LEARNING OVER FUNCTIONS, DISTRIBUTIONS AND DYNAMICS VIA STOCHASTIC OPTIMIZATION

A Dissertation
Presented to
The Academic Faculty

By

Bo Dai

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology

August 2018

LEARNING OVER FUNCTIONS, DISTRIBUTIONS AND DYNAMICS VIA STOCHASTIC OPTIMIZATION

Approved by:

Dr. Le Song, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Hongyuan Zha
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Byron Boots
School of Interactive Computing
Georgia Institute of Technology

Dr. Guanghui Lan
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Arthur Gretton
Gatsby Computational Neuroscience Unit
University College London

Date Approved: July 23, 2018

To my beloved parents,

Shuo, and Katherine

ACKNOWLEDGEMENTS

I had a wonderful journal at Georgia Institute of Technology for pursuing my Ph.D. degree in the past five years. I am deeply grateful to a group of individuals who have contributed to the completion of this thesis in many different ways.

First of all, I would like to express my foremost and deepest gratitude to my advisor, Professor Le Song, for his uncountable, visionary, inspiring, and insightful guidance during my Ph.D. period. Besides his knowledge and wisdom, his enthusiasm and research attitude always encourages me towards the elegant algorithms with state-of-the-art performances. It is my great honor and an unforgettable experience working with him. Without his relentless support and extraordinary supervision, I could not achieve what I have right now.

Moreover, I would like to extend my sincere appreciation to my committee, Byron Boots, Arthur Gretton, Guanghui Lan and Hongyuan Zha, for their generous efforts and valuable suggestions to this dissertation.

During my Ph.D. period, I have been extremely fortunate to collaborate with many great minds, Animashree Anandkumar, Maria-Florina Balcan, Byron Boots, Niao He, Sanjiv Kumar, Lihong Li, Alexander Smola, Masashi Sugiyama, Lin Xiao, and Hongyuan Zha, who have prominent impact on my research with their outstanding expertise and profound vision. My research has also profited significantly from interacting with a group of awesome coauthors. I am grateful to all of them: Jianshu Chen, Hanjun Dai, Ruiqi Guo, Xingguo Li, Yingyu Liang, Weiyang Liu, Zhe Liu, Gang Niu, Yunpeng Pan, Albert Shaw, Bo Xie, Zenglin Xu, and Tuo Zhao. I have learned a lot from the discussions with Gang Niu, Yingyu Liang, and Tuo Zhao. I will never forget the collaboration experiences with Hanjun Dai, Bo Xie, and Weiyang Liu for trying to catch each deadline.

I must thank to all my wonderful friends, classmates and colleagues at Georgia Tech, Binghong Chen, Shang-Tse Chen, Nan Du, Rundong Du, Mehrdad Farajtabar, Amrita Gupta, Elias Khalil, Jiajia Li, Shuang Li, Woosang Lim, Xing Liu, Yongchao Liu, Rakshit Trivedi, Yichen Wang, Yanming Zhang, Yuyu Zhang, and many others, for every enjoyable moment.

Finally and the most importantly, I feel strongly indebted to the unconditional love and

support from my parents. I would never have gone so far without their selfless sacrifices. I would express my special thanks to my wife, Shuo, for her endless support, lasting company, cheerful encouragement and her patience to such an always busy husband during the whole journal. My family, Shuo, Katherine, and my parents, are forever the best of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this dissertation, we reformulate various learning problems by leveraging the optimization with integral operators in the infinite-dimensional space view. Based on such a new understanding, we propose a unified framework for machine learning, which covers plenty of learning problems. It can potentially lead to better algorithms that bypass the difficulties and drawbacks in existing algorithms for these problems. Particularly, this dissertation will demonstrate the benefits of the framework for three learning problems, each of which contributes to one of the representative topics in machine learning community:

- **Learning over Functions** Being motivated from scaling up the kernel machines, we reformulate the learning problems for most of the kernel machine as a special case of the proposed framework, *i.e.*, optimization over function inputs. The new formulation avoids the operations on kernel matrices and leads to a scalable algorithm for many kernel methods. We provide the sample complexity and convergence analysis for the algorithm and justify its empirical performance on extensive experiments.

- **Learning over Distributions** Targeting on the provable algorithm for the Bayesian inference, we recast the Bayesian inference as a special case of the proposed framework, *i.e.*, optimization over distributions in the density space. It leads to a flexible and efficient inference algorithm for graphical models. We provide the convergence guarantee for the proposed algorithm and evaluate its performance for several probabilistic models on large-scale datasets.

- **Learning over Dynamics** We abstract a generic class of learning problems from many machine learning tasks, *e.g.*, invariance learning and reinforcement learning, as an important instance of the proposed framework. It learns a function with conditional distributions or dynamics as inputs. We derive a new min-max reformulation

for such problems, leading to the novel algorithm which can be tailored to many particular tasks. We provide both theoretical and empirical justification of the proposed algorithms.

Besides these concrete examples, the proposed framework can also shed light on many other learning problems, *e.g.*, undirected graphical model learning, meta learning, and so on. The optimization with the integral operator in function space view offers us a principled way to develop flexible models, design efficient algorithms, and provide rigorous guarantees. We believe it will benefit wider spectrum of machine learning applications.

Keywords: Nonparametric method, Stochastic optimization, Reproducing kernel Hilbert space (RKHS), Functional gradient, Bayesian inference, Monte-Carlo approximation, Fenchel's duality, Saddle-point problem, Reinforcement learning, Markov decision process, Bellman equation

CHAPTER 1
INTRODUCTION

The learning objectives, model representations and learning algorithms are important components of machine learning methods. Most machine learning methods are constructed by composing these three components appropriately. The learning objective is usually designed for particular tasks, *e.g.*, evidence lower bound in Bayesian inference [1] and the temporal-difference error in reinforcement learning [2]. The representation of the model is decided by the assumptions on the hypothesis space with the mechanisms to handle the structures in inputs, *e.g.*, the latent Dirichlet allocations model for text data [3], the RKHS functions for distributional data [4, 5], the convolution neural networks (CNN) [6] and residual networks [7] for 2D images, and the recurrent neural networks for sequential data [8]. Finally, the development for efficient and scalable learning algorithms usually depends on the choice of both the learning objectives and the function representations, *e.g.*, the sequential minimal optimization for SVM [9] and the back-propagation for CNN [10]. As we can see, to construct successful machine learning methods that are naturally fit to different problems with different targets and inputs, one should consider these three components together in a principled way.

This dissertation aims for developing a unified learning framework for such purpose. The heart of this framework is the optimization with integral operators in infinite-dimensional spaces. Such an integral operator representation view in the proposed framework provides us an abstract tool for considering these three components together for plenty of machine learning tasks and will lead to efficient algorithms equipped with flexible representations achieving better approximation ability, scalability and statistical properties.

We mainly investigate several motivated machine learning problems, *i.e.*, kernel methods [11], Bayesian inference [12], invariance learning, policy evaluation and policy optimization in reinforcement learning [13, 14], as the special cases of the proposed framework with different instantiations of the integral operator. These instantiations result in the learning

problems with inputs as functions, distributions, and dynamics. The corresponding algorithms are derived to handle the particular integral operators via efficient and provable stochastic approximation by exploiting the particular structure properties in the operators. The proposed framework and the derived algorithms are deeply rooted in functional analysis, stochastic optimization, nonparametric method, and Monte Carlo approximation, and contributed to several sub-fields in machine learning community, including kernel methods, Bayesian inference, and reinforcement learning.

We believe the proposed framework is a valuable tool for developing machine learning methods in a principled way and can be potentially applied to many other scenarios, *e.g.*, kernel exponential families estimation and meta-learning.

1.1 Thesis Structure

The rest of this dissertation is organized into six chapters.

Chapter 2: A Unified Learning Framework We first provide the background knowledge, including the Hilbert space and its properties, that is needed for establishing the whole framework. Next, we introduce the learning framework based on the optimization with integral operators in infinite-dimension function spaces perspective. The framework unifies many representative machine learning problems, *e.g.*, kernel methods, Bayesian inference, and reinforcement learning, into one complete picture, while it also preserves the useful properties for us to design better algorithms in terms of approximation ability, statistical, computational and memory complexity for particular problems.

Chapter 3: Doubly Stochastic Gradient Descent in RKHS In this chapter, we specialize the kernel methods from the framework with a novel representation, so that we can scale up the kernel machines to millions of samples. The bottleneck of kernel machines for large-scale datasets is the computation and memory cost of the kernel matrix. By reformulating the learning of kernel machines as an optimization in the dual function space of the corresponding RKHS through the framework, we can avoid the explicit operations on kernel matrices, and thus, bypass the well-known bottleneck. The resulted *doubly stochastic*

functional gradient algorithm approximates the functional gradient stochastically from two randomness sources, *i.e.*, the training data and the dual random features associated with the kernel. We prove that the proposed algorithm converges in the optimal rate and conduct extensive empirical experiments, demonstrating the scalability of the proposed algorithm for kernel machines on millions of samples.

Chapter 4: Particle Mirror Descent in Density Space In this chapter, we discuss the Bayesian inference problem, which can be recast as another special case of the proposed framework, *i.e.*, optimization in the density space over distribution inputs. By such a reformulation, we propose a simple yet provable algorithm, *particle mirror descent*, to iteratively approximate the posterior density. The proposed algorithm incorporates the particles to represent the density nonparametrically in the stochastic functional mirror descent updates, and thus, bypasses the restrictions in the approximation ability in vanilla variational inference algorithms due to the prefixed parametrization. We provide the convergence guarantee of the proposed algorithm and demonstrate its empirical performances in several probabilistic models on large-scale datasets.

Chapter 5: Dual Embedding for Conditional Integral Operator In this chapter, we investigate the learning over dynamics inputs problem, which can be derived from the proposed framework by plugging the conditional integral operator. This problem is motivated from invariance learning, policy evaluation in Markov decision processes (MDPs), and events prediction problems. We propose the *dual embedding* to bypass the difficulty in estimating the conditional expectation when we only have limited samples or in the extreme case only one sample for each conditional distribution in the dynamics. We design an efficient learning algorithm, *Embedding-SGD*, with convergence guarantees for such problems. Finally, our numerical experiments on both synthetic and real-world datasets show that the proposed approach can significantly improve over the existing algorithms on invariance learning and policy evaluation in MDPs.

Chapter 6: Dual Embedding for Smoothed Bellman Error In this chapter, motivated by policy optimization in MDPs, we mainly discuss a difficult generalization of the

optimization problem in Chapter 5. By the property of the smoothed Bellman optimality equation, we derive a specific surrogate loss for policy optimization, which can be reduced to the learning problem in the framework with the conditional integral operator. We apply the dual embedding technique for a new reinforcement learning algorithm. The proposed algorithm is proved convergent, even with nonlinear function approximation on off-policy samples. We compare the proposed algorithm with the current existing reinforcement learning algorithms on the benchmarks, and demonstrate its convergence and sample efficiency.

Conclusion We summarize the achievements in this thesis. We also discuss some potential future directions to further complete the proposed unified learning framework, including undirected graphical model learning and meta-learning.

1.2 Thesis Contribution

This dissertation includes the conceptual, algorithmic and theoretical contributions:

1. A unified framework for machine learning from the integral operator view, which unifies several representative learning problems as special cases.
2. A scalable algorithm, *doubly stochastic gradient descent*, for general kernel methods.
3. A provable algorithm, *particle mirror descent*, for approximate Bayesian inference on probabilistic models.
4. An efficient algorithm, *embedding-SGD*, for learning over dynamics problems, *e.g.*, invariant learning, policy evaluation and so on.
5. A convergent algorithm, *smoothed Bellman error embedding*, for off-policy reinforcement learning with nonlinear function approximation.
6. Theoretical analysis for characterizing the behaviors of the proposed algorithms.

These contributions demonstrate the importance of the proposed framework as a principled way for machine learning method design. Besides deriving useful algorithms for concrete problems, the theoretical analysis is also of independent interest which is potentially applicable for justifying the other existing related algorithms.

CHAPTER 2
A UNIFIED LEARNING FRAMEWORK

In this chapter, we will first introduce the necessary background of the framework. Then, we present our unified learning framework, which is based on optimization with integral operators in infinite-dimension spaces, and discuss the important special cases of the framework, including kernel methods, Bayesian inference, invariance learning and reinforcement learning with the inputs as functions, distributions, and dynamics, respectively.

2.1 Preliminary

The proposed framework is built up based on the integral operator view of learning problems. In this section, we mainly discuss the Hilbert space and its properties, which will be important for developing the framework.

2.1.1 Hilbert Space

We introduce the Hilbert space starting from the definition of the inner product space,

Definition 1 (Inner product) *Let Ξ denotes a vector space, an inner product on Ξ is a function $\langle \cdot, \cdot \rangle : \Xi \times \Xi \to \mathbb{R}$ that satisfies following three axioms for $x, y, z \in \Xi$ and $\alpha, \beta \in \mathbb{R}$:*

- **linearity** $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$.
- **symmetry** $\langle x, y \rangle = \langle y, x \rangle$.
- **positive-definiteness** $\langle x, x \rangle \geqslant 0$ *and* $\langle x, x \rangle = 0$ *if and only if* $x = 0$.

The inner product space is a vector space with inner product. In fact, the inner product space is also a normed space, *i.e.*,

Theorem 2 *Let $(\Xi, \langle \cdot, \cdot \rangle)$ be an inner product space and $\|x\| := \sqrt{\langle x, x \rangle}$, then, $\|\cdot\|$ is a norm on Ξ and $(\Xi, \|\cdot\|)$ is a normed space.*

Based on the understanding of the inner product space and the induced norm, the Hilbert space is defined below

Definition 3 (Hilbert space) *Let $(\Xi, \langle\cdot,\cdot\rangle)$ be an inner product space, $(\Xi, \langle\cdot,\cdot\rangle)$ is a Hilbert space if the space is complete w.r.t. the metric generated by the norm induced by the inner product.*

One of the important examples of Hilbert spaces is $L_2(\mathcal{X}, \mu)$, which is the space of square Lebesgue integrable functions $\Xi = \{f : \mathcal{X} \to \mathbb{R}\}$ with the inner product defined as $\langle f, g\rangle := \int_\Xi f(x)g(x)d\mu(x)$. Another important class of Hilbert spaces is the reproducing kernel Hilbert space as introduced in next section.

2.1.2 Reproducing Kernel Hilbert Space

The reproducing kernel Hilbert space (RKHS) is a special Hilbert space. The theory was developed by [15] and introduced to machine learning community by [16, 17]. It is formally defined below,

Definition 4 (Reproducing kernel Hilbert space [16, 17]) *Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$, then $\mathcal{H}$ is a reproducing kernel Hilbert space endowed with the inner product $\langle\cdot,\cdot\rangle_\mathcal{H}$ if there exists a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that*

- **membership** *for every $x \in \mathcal{X}$, the function $k(x,\cdot) \in \mathcal{H}$.*
- **reproducing property** *for $\forall f \in \mathcal{H}$ and $\forall x \in \mathcal{X}$, we have $f(x) = \langle f, k(x,\cdot)\rangle_\mathcal{H}$.*

In L_2 space, we have $f(x) = \int f(x')\delta(x - x')\,d\mu(x')$. However, the L_2 is not a RKHS since δ-function does not belong to L_2 space. The kernel function is analogues of δ-function, but itself belongs to $\mathcal{H}$.

By the definition, the RKHS is simply a Hilbert space of functions spanned by $k(\cdot,\cdot)$, *i.e.*, $\mathcal{H} = \overline{\text{span}\{k(x,\cdot)|x \in \mathcal{X}\}}$. We can derive some basic properties straightforwardly,

Theorem 5 *1 If k is a reproducing kernel for $\mathcal{H}$, then for $\forall x \in \mathcal{X}$ and $\forall f \in \mathcal{H}$, $|f(x)| \leqslant \sqrt{k(x,x)}\,\|f\|_\mathcal{H}$.*

2 If $\mathcal{H}$ is a RKHS, then the convergence in $\mathcal{H}$ w.r.t. $\|\cdot\|_\mathcal{H}$ implies pointwise convergence.

There is an one-to-one correspondence between k and its RKHS $\mathcal{H}$, *i.e.*,

Theorem 6 (Moore-Aronszajn [18]) *Given a positive definite function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ we can construct a unique RKHS with k as its reproducing kernel, and vice versa.*

Due to the positive definiteness of the kernel function k, we can decompose the kernel function via eigenfunctions justified by the following theorem,

Theorem 7 (Mercer decomposition [19]) *Let k is positive definite, we can expand $k(\cdot, \cdot)$ as*

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x'),$$

where $\{\psi_i\}_{i=1}^{\infty}$ are orthonormal and $\lambda_i \geqslant 0$.

Instead of the Mercer's decomposition, we can also decompose the kernel in alternative ways,

Theorem 8 (*e.g.*,[20]; [21]) *If $k(x, x')$ is a positive definite (PD) kernel, then there exists a set Ω, a measure $\rho(\omega)$ on Ω, and random feature $\phi_\omega(x) : \mathcal{X} \mapsto \mathbb{R}$ from $L_2(\Omega, \rho)$, such that $k(x, x') = \int_\Omega \phi_\omega(x) \phi_\omega(x') d\rho(\omega)$.*

It should be mentioned that the (ϕ_ω, ρ) may not be unique.

Both the Mercer decomposition and the expectation decomposition of the kernel function, in Theorem 7 and Theorem 8, respectively, can be used for representing the functions in $\mathcal{H}$ as we will introduce in next section.

2.1.3 Integral Operator

In this section, we will use the integral operator to derive the representation of the functions $f \in \mathcal{H}$. The integral operator is a useful tool for studying the properties of RKHS [22, 23]. Let $\mathcal{K} : L_2(\mu) \to L_2(\mu)$ defined as

$$(\mathcal{K}f)(x) := \int_{\mathcal{X}} f(x') k(x', x) d\mu(x').$$

Since $\mathcal{K}f$ is a linear combination of kernel functions $k(x', \cdot)$, $\mathcal{K}f \in \mathcal{H}$.

Self-adjoint square root of $\mathcal{K}$ When the $\mathcal{H}$ is dense in $L_2(\mu)$, the unique positive self-adjoint square root of $\mathcal{K}$, denoted as $\mathcal{K}^{\frac{1}{2}}$, is a bijection from $L_2(\mu)$ to $\mathcal{H}$ [24, 23]. Therefore, for $\forall h \in \mathcal{H}$, $\exists f \in L_2(\mu)$, we can represent $h = \mathcal{K}^{\frac{1}{2}} f$ and $\|h\|_{\mathcal{H}} = \|f\|_2$. Moreover, $\forall f, g \in \mathcal{H}$, we have

$$\langle f, g \rangle_{\mathcal{H}} = \left\langle \mathcal{K}^{-\frac{1}{2}} f, \mathcal{K}^{-\frac{1}{2}} g \right\rangle_2 .$$

By Theorem 7 and the fact $\mathcal{H}$ is dense in $L_2(\mu)$, the λ_i $i = 1, \ldots, \infty$ are strict positive. Then, we can use the eigendecomposition to represent the $\|\cdot\|_{\mathcal{H}}$, *i.e.*,

$$\|f\|_{\mathcal{H}}^2 = \left\| \mathcal{K}^{-\frac{1}{2}} f \right\|_2^2 = \sum_{i=1}^{\infty} \lambda_i^{-1} \langle f, \psi_i \rangle_2^2 .$$

Square root of $\mathcal{K}$ Define the bounded linear operator $\mathcal{Q} : L_2(\rho) \to L_2(\mu)$ as

$$(\mathcal{Q}g)(x) := \int_{\Omega} g(\omega) \phi_\omega(x) d\rho(\omega), \tag{2.1}$$

we have a unique adjoint operator $\mathcal{Q}^* : L_2(\mu) \to L_2(\rho)$ as $\langle g, \mathcal{Q}^* f \rangle_2 = \langle \mathcal{Q}g, f \rangle_2$ and expressed as

$$(\mathcal{Q}^* f)(\omega) = \int_{\mathcal{X}} f(x) \phi_\omega(x) \, d\mu(x).$$

Since $\mathcal{Q}$ is a linear operator, it defines a bijection between the orthogonal of its null space, $Ker(\mathcal{Q})^{\perp}$, to its image, $Im(\mathcal{Q})$, which implies the inverse operator $\mathcal{Q}^{-1} : Im(\mathcal{Q}) \to Ker(\mathcal{Q})^{\perp}$. As shown in [25, 23], $Im(\mathcal{Q})$ is exact the RKHS $\mathcal{H}$ with kernel $k(x, x') = \int_\omega \phi_\omega(x) \phi_\omega(x') \, d\rho(\omega)$. Then, $\forall f \in \mathcal{H}$, it can be represented as $f = \mathcal{Q}h$ for some $h \in Ker(\mathcal{Q})^{\perp}$, and $\forall f, g \in Im(\mathcal{Q})$,

$$\langle f, g \rangle_{\mathcal{H}} = \left\langle \mathcal{Q}^{-1} f, \mathcal{Q}^{-1} g \right\rangle_2 , \quad \|f\|_{\mathcal{H}} = \|h\|_2 .$$

We can also check the decomposition $\mathcal{K} = \mathcal{Q}\mathcal{Q}^*$ by simply applying the Fubini's theorem. Since the (ϕ_ω, ρ) is not unique, we may have many possible $\mathcal{Q}$ corresponding to different $Im(\mathcal{Q})$.

2.1.4 Functional Gradient

The first-order algorithms in the function space is one key component for deriving the algorithm from the framework. We briefly introduce the functional gradient on which the first-order algorithms is built.

The functional gradient relates a change in terms of the functional to a change in function on which the functional depends. Denote the functional as $L(f)$, the functional gradient $\nabla L(f)$ is defined as the linear term in the change of the objective after we perturb f by ϵ in the direction of g, *i.e.*,

$$L(f + \epsilon g) = L(f) + \epsilon \langle \nabla L(f), g \rangle + O(\epsilon^2). \tag{2.2}$$

where $\langle \cdot, \cdot \rangle$ can be adapted with different definitions in different function spaces. For instance, in RKHS where we have inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, applying the above definition, we have $\nabla f(x) = \nabla \langle f, k(x, \cdot) \rangle_{\mathcal{H}} = k(x, \cdot)$ and $\nabla \|f\|_{\mathcal{H}}^2 = \nabla \langle f, f \rangle_{\mathcal{H}} = 2f$.

Given the definition of the functional gradient, we can generalize the first-order optimization method and its stochastic version to function spaces, *e.g.*, stochastic gradient descent and stochastic mirror descent, while still keep the convergence guarantees. We will see in the following chapters how to apply these generalized algorithms in function spaces for particular learning problems instantiated from the proposed framework.

2.2 Learning as Optimization with Integral Operator

With the introduced background knowledge, we present our unified framework built on the integral operator view. By instantiating different integral operators in the framework, we recast various machine learning problems into the unified view, which will help us to design novel algorithms bypassing the drawbacks in existing algorithms.

In general, we consider the learning tasks as searching for an element in a Hilbert space $\mathcal{F}$, *e.g.*, RKHS, density space, L_2 space and so on, by minimizing a functional $L(\cdot) : \mathcal{F} \to \mathbb{R}$.

Specifically, we have the optimization as

$$\min_{f\in\mathcal{F}} \; L\left(f\right) := \mathbb{E}_s\left[J\left(\mathcal{T}f, s\right)\right] + \nu G(f), \tag{2.3}$$

where $\mathcal{T} : \mathcal{F} \to \mathcal{G}$ denotes some integral operator that maps functions in $\mathcal{F}$ to another space $\mathcal{G}$, $\mathbb{E}_s\left[\cdot\right]$ denotes the expectation over some measure $p(s)$, $J\left(\cdot,\cdot\right) : \mathcal{F}\times\mathcal{S} \to \mathcal{Y}$ denotes the loss function which is convex w.r.t. the first argument and $G\left(\cdot\right)$ denotes an optional convex regularization, $\nu \geqslant 0$ is the regularization parameter. The framework covers various machine learning models by adapting different integral operators $\mathcal{T}$, ranging from kernel methods, Bayesian inference, invariance learning, to reinforcement learning with inputs as functions, distributions, and dynamics, respectively.

2.2.1 Learning Over Functions

If we instantiate the integral operator $\mathcal{T}$ in Eq (2.3) as $\mathcal{Q} : L_2\left(\rho\right) \to \mathcal{H}$ defined in Section 2.1 and $G\left(\cdot\right)$ as $\left\|\cdot\right\|_{\mathcal{H}}$, we obtain

$$\min_{g\in L_2(\rho)} \; \mathbb{E}_{x,y}\left[J\left(\mathcal{Q}g, (x,y)\right)\right] + \nu\left\|g\right\|_2^2 = \mathbb{E}_{x,y}\left[\ell\left(\mathbb{E}_{\omega\sim\rho(\omega)}\left[\phi_\omega\left(x\right)g\left(\omega\right)\right], y\right)\right] + \nu\left\|g\right\|_2^2. \tag{2.4}$$

where $\ell\left(\mathbb{E}_{\omega\sim\rho(\omega)}\left[\phi_\omega\left(x\right)g\left(\omega\right)\right], y\right) := J\left(\mathcal{Q}g, (x,y)\right) : \mathbb{R}\times\mathbb{R} \to \mathcal{Y}$ denotes a convex loss and $\mathbb{E}_{x,y}\left[\cdot\right]$ denotes the expectation over distribution $p(x,y)$. By such a representation, we can understand the learning problems corresponding to Eq (2.4) as learning a function $g\left(\omega\right)$ with the inputs as functions $\phi_\omega\left(x\right)$ over Ω. Therefore, we refer to the learning problem (2.4) as learning over functions.

Recall that the integral operator $\mathcal{Q}$ induces a mapping from $L_2\left(\rho\right)$ to RKHS $\mathcal{H}$, *i.e.*, $\mathcal{Q}g \in \mathcal{H}$ for $g \in L_2\left(\rho\right)$ as we introduced in Section 2.1.3, the optimization (2.4) represents a wide range of kernel methods, including kernel ridge regression, kernel SVM, kernel logistic regression, kernel density ratio estimation and so on, as a special case of the proposed framework with different $J(\cdot,\cdot)$.

This integral operator view of kernel methods in Eq (2.4) derived from the proposed framework avoids the explicit operations on the kernel and leads to a novel algorithm,

which is able to scale up a general class of kernel machines to millions of data. The proposed algorithm achieves a delicate balance between computational, memory and statistical complexity. We will discuss the details in Chapter 3.

2.2.2 Learning Over Distributions

If we instantiate the integral operator in Eq (2.3) as $(\mathcal{T}q)(x) := \langle \log p(x|\theta), q(\theta) \rangle : \mathcal{P} \to \mathbb{R}$ where $\langle \cdot, \cdot \rangle$ denotes the inner product in L_2 space, $\log p(x|\theta)$ denotes the log-likelihood, $G(q)$ as $KL(q||p) := \left\langle q, \log \frac{q}{p} \right\rangle$ with $p(\theta)$ as prior, $\nu = \frac{1}{N}$, and $J(\mathcal{T}q, x) := -(\mathcal{T}q)(x)$ as the linear loss, we obtain

$$\min_{q \in \mathcal{P}} -\widehat{\mathbb{E}}_x [(\mathcal{T}q)(x)] + \frac{1}{N} KL(q||p) = \frac{1}{N} \sum_{i=1}^{N} \int q(\theta) \log p(x|\theta) \, d\theta + \frac{1}{N} \int q(\theta) \log \frac{q(\theta)}{p(\theta)} d\theta, \tag{2.5}$$

where $\widehat{\mathbb{E}}_x [\cdot]$ denotes the empirical expectation over the training dataset $\mathcal{D} = \{x_i\}_{i=1}^N$. It has been shown that the solution to Eq (2.5), denoted as $q^*(\theta)$, is exactly the posterior from Bayes' rule [26, 1], *i.e.*,

$$q^*(\theta) = \frac{p(\theta) \prod_{i=1}^N p(x_i|\theta)}{Z}, \tag{2.6}$$

where $Z = \int p(\theta) \prod_{i=1}^N p(x_i|\theta) d\theta$. Therefore, we can reformulate the Bayesian inference as a special case of the proposed framework. With such instantiation, the learning problem corresponding to Bayesian inference can be understood as learning a density with the inputs as distributions $p(x|\theta)$. Therefore, we refer the optimization (2.5) as learning over distributions.

By the optimization view of Bayesian inference in Eq (2.5) specialized from the general framework, we can bypass the notorious difficulty in Bayesian inference, *i.e.*, the intractability of the partition function Z in Eq (2.6). We propose a novel algorithm for Bayesian inference, which is flexible to approximate arbitrary smooth posteriors and still guaranteed convergence with finite-step rate. We will discuss the details in Chapter 4.

2.2.3 Learning Over Dynamics

We can induce even richer structure into the integral operator to achieve a sub-class of learning problems which includes invariance learning, reinforcement learning and events prediction as special cases. Specifically, we instantiate the integral operator in Eq (2.3) as

$$\left(\mathcal{T}f\right)(x) := \langle p(z|x), f(z,x)\rangle = \mathbb{E}_{z|x}\left[f(x,z)\right] : \mathcal{F}\left(\mathcal{X}\times\mathcal{Z}\right) \to \mathcal{F},$$

where $\mathbb{E}_{z|x}\left[\cdot\right]$ denotes the conditional expectation, and $G\left(\cdot\right)$ as some norms, which will be selected for particular problems. Then, the optimization (2.3) will be specialized as

$$\min_{f\in\mathcal{F}} \ \mathbb{E}_{x,y}\left[J\left(\mathbb{E}_{z|x}\left[f(\cdot,z)\right],(x,y)\right)\right] + \nu G\left(f\right). \tag{2.7}$$

By such representation, we can understand the learning problems corresponding to Eq (2.7) as learning a function with the inputs as conditional distribution $p(z|x)$. Since the conditional distribution fully characterizes the transition of a dynamics, we refer to the learning problem in (2.7) as learning over dynamics.

The problem of learning over dynamics appears in many different tasks. For example, the goal of invariance learning is to estimate a function which minimizes the expected risk while at the same time preserving consistency over a group of operations $g = \{g_j\}_{j=1}^{\infty}$. [27] shows that this can be accomplished by solving the following optimization problem,

$$\min_{f\in\mathcal{H}} \ \mathbb{E}_{x,y}[\ell(\mathbb{E}_{z|x\sim\mu(g(x))}[f(z)],y)] + \nu\|f\|_{\mathcal{H}}^2 \tag{2.8}$$

where $\mathcal{H}$ is the RKHS and $\ell(\mathbb{E}_{z|x\sim\mu(g(x))}[f(z)],y) := J\left(\left(\mathcal{T}f\right),(x,y)\right) : \mathbb{R}\times\mathbb{R}\to\mathcal{Y}$ denotes a convex loss. The optimization is clearly a special case of Eq (2.7). The policy evaluation is another example. Given a policy $\pi(a|s)$ which is a distribution over action space condition on current state s, the goal is to estimate the value function $V^{\pi}(\cdot)$ over the state space by minimizing the mean-square Bellman error [28, 29]:

$$\min_{V^{\pi}\in\mathcal{H}} \ \mathbb{E}_{s}\left[\left(\mathbb{E}_{s',a|s}\left[R(s,a)+\gamma V^{\pi}(s')-V^{\pi}(s)\right]\right)^2\right] + \nu\left\|V\right\|_{\mathcal{H}}^2, \tag{2.9}$$

where $\mathbb{E}_s[\cdot]$ denote the expectation over $p_0(s)$, $\mathbb{E}_{s',a|s}$ denote the conditional expectation over $p(s'|a,s)\pi(a|s)$ and $R(s,a):\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ is a reward function with $\mathcal{S}$ and $\mathcal{A}$ denoting the state and action space, respectively, and $\gamma\in(0,1)$ is the discount factor. It is a special case of the Eq (2.7) by setting $(x,y)=s, z=(s',a)$, $f(s,a,s')=R(s,a)+\gamma V^\pi(s')-V^\pi(s)$, $J(\cdot,\cdot)$ as square loss, and $G(\cdot)$ as $\|V\|^2_{\mathcal{H}}$.

These learning problems become very challenging when we only have limited samples or in the extreme case only one sample from each conditional distribution. Based on the saddle-point formulation from the optimization view, we design a novel algorithm, which is efficient to address this sampling challenge. The proposed algorithm can be tailored to the invariance learning, policy evaluation, and other special cases. The details are explained in Chapter 5.

Motivated by policy optimization in reinforcement learning, we are also interested in a difficult generalization of the learning over dynamics problem, *i.e.*, estimating optimal value function $V(s)$ by minimizing mean-square Bellman optimality error:

$$\min_{V\in\mathcal{H}}\mathbb{E}_s\left[\left(\max_{\pi(a|s)\in\mathcal{P}_\mathcal{A}}\mathbb{E}_{s'|a,s}\left[R(s,a)+\gamma V(s')-V(s)\right]\right)^2\right]. \tag{2.10}$$

The operator in Eq (2.10) now involves a max-operator over policies which are distributions, which induces extra complicatedness. We derive a surrogate of Eq (2.10), which can be reduced to the optimization (2.7), by exploiting the property of smoothed Bellman error. Based on the technique developed in Chapter 5, we propose a new reinforcement learning algorithm which is proved convergent on off-policy samples with nonlinear function approximation. We will discuss the details in Chapter 6.

2.3 Summary

To conclude this chapter, we remark that many machine learning methods have been proposed for different learning tasks. The major differences of different methods lie in three aspects: **i)**, the learning target; **ii)**, the representation of the functions and the input forms; **iii)**, the algorithms for optimizing the objectives. In fact, to design a successful machine learning method for a particular problem, we should consider these three components to-

gether to achieve the best theoretical property and empirical performance.

By exploiting the optimization with the integral operator view, we propose a framework integrating these three components as a whole. It is relatively abstract to contain many representative learning methods, *e.g.*, kernel methods, Bayesian inference, invariance learning and reinforcement learning, as special cases, while still preserves necessary structures in integral operators which will be important for designing practical algorithms. The remainder of the dissertation will be separated into three parts, each of which corresponds to one of the representative problems in following chapters.

PART I: LEARNING OVER FUNCTIONS

As we show in Section 2, due to the expectation decomposition of kernel function in Theorem 8 and the equivalence of the RKHS and image of the square root of integral operator with kernel $k(\cdot,\cdot)$, most of the kernel methods can be instantiated from the framework (2.3) as below,

$$\min_{g\in L_2(\rho)} \mathbb{E}_{x,y}\left[\ell\left(\mathbb{E}_{\omega\sim\rho(\omega)}\left[\phi_\omega(x)\, g(\omega)\right], y\right)\right] + \nu\|g\|_2^2. \tag{2.11}$$

The optimization can be understood as treating the inputs as $\phi_\omega(x) \in L_2(\Omega,\rho)$ and searching for a linear function $g(\omega) \in L_2(\Omega,\rho)$. In other words, most of the kernel methods, *e.g.*, kernel support vector machines, kernel logistic regression, kernel ridge regression, kernel quantile regression, kernel novelty detection [30], kernel density ratio estimation [31], and so on, can be recast as *learning over functions* from the proposed framework with the special integral operator.

The most significant advantage of such a representation of kernel methods is that we can bypass the explicit computation and memory cost of kernel, which is the major bottleneck to apply the kernel methods to millions of training data. Therefore, we design an efficient algorithm, *doubly stochastic gradient descent*, to scale up the kernel methods through the integral operator view in Chapter 3.

CHAPTER 3

DOUBLY STOCHASTIC GRADIENT DESCENT IN RKHS

Kernel methods are nonparametric methods which provide a flexible modeling tool that captures complex dependencies and structures automatically without task-dependent design. They have demonstrated empirical success in many medium scale applications, including regression, classification, dimension reduction, visualization and dynamic systems [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 16, 43, 44, 45, 46, 47, 48, 49]. Moreover, the most attractive benefit of kernel methods are their nice theoretical properties: since the learning procedures of most kernel methods can be formulated as convex optimizations, we can achieve the global optima, and thus, statistical convergence guarantees can be provided [50, 51, 52, 53, 54]. However, scaling up kernel methods to large-scale datasets with millions of data is very challenging: kernel methods typically work with the kernel matrices which are quadratic in the number of samples and imposing huge burden in both computation and storage.

Rewriting the kernel methods through the integral operator from proposed framework, *i.e.*,

$$\min_{g \in L_2(\rho)} \mathbb{E}_{x,y} \left[\ell \left(\mathbb{E}_{\omega \sim \rho(\omega)} \left[\phi_\omega (x) \, g (\omega) \right], y \right) \right] + \nu \left\| g \right\|_2^2, \tag{3.1}$$

we eliminate the existence of the kernel, providing the opportunity to scale up the kernel methods. The reformulation inspires the *doubly stochastic gradient descent* algorithm. The algorithm makes *two folds* stochastic approximations to the functional gradient, one using random training points and another using random features, *i.e.*, $\phi_\omega (\cdot)$, associated with the operator, and then descending using this noisy functional gradient. The resulted algorithm is simple and flexible, and can be applied in the streaming setting. We show that a function learned by this procedure after t iterations converges to the optimal function in the original RKHS in rate $O(1/t)$ with $O(1/\sqrt{t})$ generalization error. In both synthetic and real-world datasets, the proposed algorithm compares favorably to the other existing algorithms for kernel methods in terms of accuracy and computational and memory cost.

3.1 Introduction

The general perception is that kernel methods are not scalable. When it comes to large-scale nonlinear learning problems, the methods of choice so far are neural nets where theoretical understanding remains incomplete. The bottleneck in scaling up kernel methods is the storage and computation of the kernel matrix, K, which is usually dense. Storing the matrix requires $O(n^2)$ space, and computing it takes $O(n^2 d)$ operations, where n is the number of data points and d is the dimension. There have been many great attempts to scale up kernel methods, including efforts from numerical linear algebra, functional analysis, and numerical optimization perspectives.

A common numerical linear algebra approach is to approximate the kernel matrix using low-rank factors, $K \approx A^\top A$, with $A \in \mathbb{R}^{r \times n}$ and rank $r \leqslant n$. This low-rank approximation usually requires $O(nr^2 + nrd)$ operations, and then subsequent kernel algorithms can directly operate on A. Many works, such as greedy basis selection techniques [55], Nyström approximation [56] and incomplete Cholesky decomposition [57], all followed this strategy. In practice, one observes that kernel methods with approximated kernel matrices often result in a few percentage of losses in performance. In fact, without further assumption on the regularity of the kernel matrix, the generalization ability after low-rank approximation is typically of the order $O(1/\sqrt{r} + 1/\sqrt{n})$ [58, 59], which implies that the rank needs to be nearly linear in the number of data points! Recently, the leverage score sampling has been introduced to reduce the error in the low-rank approximation of the kernel matrix [60, 61]. However, extra computation will be needed for obtaining the leverage score. The computational cost for the exact leverage score is in the same order as solving the original problem, therefore, we need one more approximation stage for large-scale problems! Moreover, the theoretical analysis in [60, 61] is only valid for kernel ridge regression. The theoretical property of this approximation for the kernel machines with other losses is not clear. In sum, for general kernel machines with arbitrary loss functions as we discussed in Section 3.3, in order for kernel methods to achieve the best generalization ability, the low-rank approximation based approaches quickly become impractical for big datasets due to their $O(n^3 + n^2 d)$ preprocessing time and $O(n^2)$ memory requirement.

Random feature approximation is another popular approach for scaling up kernel methods [62, 63]. Instead of approximating the kernel matrix, the method directly approximates the kernel function using explicit feature maps. The advantage of this approach is that the random feature matrix for n data points can be computed in time $O(nrd)$ using $O(nr)$ memory, where r is the number of random features. Subsequent algorithms then only operate on an $O(nr)$ matrix. Similar to low-rank kernel matrix approximation approach, the generalization ability of random feature approach is of the order $O(1/\sqrt{r}+1/\sqrt{n})$ [64, 65], which implies that the number of random features also needs to be $O(n)$. The approximation error can be reduced by adapting the sampling distribution over the random features theoretically [23]. However, the specific non-uniform distribution proposed in [23] is intractable in general, therefore, difficult to be applied in practice. Another common drawback of these two approaches is that it is not easy to adapt the solution from a small r to a large r'. Often one is interested in increasing the kernel matrix approximation rank or the number of random features to obtain a better generalization ability. Then special procedures need to be designed to reuse the solution obtained from a small r, which is not straightforward.

Another approach that addresses the scalability issue rises from optimization perspective. One general strategy is to solve the dual forms of kernel methods using coordinate or block-coordinate descent (*e.g.*, [9, 66, 67]). By doing so, each iteration of the algorithm only incurs $O(nrd)$ computation and $O(nr)$ memory, where r is the size of the parameter block. A second strategy is to perform functional gradient descent by looking at a batch of data points at a time (*e.g.*, [68, 69]). Thus, the computation and memory requirements are also $O(nrd)$ and $O(nr)$ respectively in each iteration, where r is the batch size. These approaches can easily change to a different r without restarting the optimization and has no loss in generalization ability since they do not approximate the kernel matrix or function. However, a serious drawback of these approaches is that, without further approximation, all support vectors need to be kept for testing, which can be as big as the entire training set! (*e.g.*, kernel ridge regression and non-separable nonlinear classification problems.)

In summary, there exists a delicate trade-off between computation, memory and statistics if one wants to scale up kernel methods. Inspired by the integral operator view of kernel methods, we propose a simple yet general strategy to scale up many kernel methods using

a novel concept called "*doubly stochastic functional gradients*". Our method relies on the fact that most kernel methods can be expressed as convex optimization problems over functions in reproducing kernel Hilbert spaces in the form of integral operator representation and can be solved via functional gradient descent. Our algorithm proceeds by making *two* stochastic approximations to the functional gradient, one using random training points and the other one using random features associated with the kernel. The key intuitions behind our algorithm originate from

(i) the property of stochastic gradient descent algorithm that as long as the bias of stochastic gradient estimator is comparable with the variance, the convergence of the algorithm is guaranteed; and

(ii) the property of pseudo-random number generators that the random samples can in fact be completely determined by an initial value (a seed).

We exploit these properties and enable kernel methods to achieve better balances between computation, memory and statistics.

3.1.1 Contributions

Our method interestingly combines kernel methods, functional analysis, stochastic optimization and algorithmic trick, and it possesses a number of desiderata:

Generality and simplicity Our approach applies to many kernel methods, such as kernel ridge regression, support vector machines, logistic regression, two-sample test, and many different types of kernels, such as shift-invariant kernels, polynomial kernels, general inner product kernels, and so on. The algorithm can be summarized in just a few lines of codes (Algorithm 1). For a different problem and kernel, we just need to adapt the loss function and the random feature generator.

Flexibility Different from previous uses of random features which typically prefix the number of features and then optimize over the feature weightings, our approach allows the number of random features, and hence the flexibility of the function class, to grow with the

number of data points. This allows our method to be applicable to data streaming setting, which is not possible for previous random feature approach, and achieve the full potential of nonparametric methods.

Efficient computation The key computation of our method is evaluating the doubly stochastic functional gradient, which involves the generation of the random features with specific random seeds and the evaluation of these random features on the small batch of data points. For iteration t, the computational complexity is $O(td)$.

Small memory The doubly stochasticity also allows us to avoid keeping the support vectors which becomes prohibitive in large-scale streaming setting. Instead, we just need to keep a small program for regenerating the random features, and sample previously used random feature according to pre-specified random seeds. For iteration t, the memory needed is $O(t)$ independent of the dimension of the data.

Theoretical guarantees We provide a novel and nontrivial analysis involving Hilbert space martingale and a newly proved recurrence relation, and show that the estimator produced by our algorithm, which might be outside of the RKHS, converges to the optimal RKHS function. More specifically, both in expectation and with high probability, our algorithm can estimate the optimal function in the RKHS in the rate of $O(1/t)$, which are indeed optimal [70], and achieve a generalization bound of $O(1/\sqrt{t})$. The variance of the random features, introduced during our second approximation to the functional gradient, only contributes additively to the constant in the final convergence rate. These results are the first of the kind in kernel method literature, which can be of independent interest.

Strong empirical performance Our algorithm can readily scale kernel methods up to the regimes which are previously dominated by neural networks. We show that our method compares favorably to other scalable kernel methods in medium scale datasets, and to neural networks in big datasets such as 8 million handwritten digits from MNIST, 2.3 million materials from MolecularSpace, and 1 million photos from ImageNet using convolution features. Our results suggest that kernel methods, theoretically well-grounded methods, can

potentially replace neural nets in many large scale real-world problems where nonparametric estimation are needed.

In the remainder, we will first discuss the details of the expectation decomposition of kernels for implementing the integral operator representation of RKHS function $f = \mathcal{Q}g$ concretely. We will then describe our algorithm and provide both theoretical and empirical supports.

3.2 Duality between Kernels and Random Processes

In this section, we mainly provide detailed explanation to the expectation decomposition of the symmetric positive definite kernel function for constructing $\mathcal{Q}$ concretely. As we will see, such a decomposition is not only the abstract tool to reformulate the kernel methods as a special case to our framework, but also the key component to form the scalable algorithm Algorithm 1.

There is an intriguing duality between kernels and stochastic processes as we introduced in Theorem 8. We restated as below,

Theorem 8 *If $k(x, x')$ is a positive definite (PD) kernel, then there exists a set Ω, a measure $\rho(\omega)$ on Ω, and random feature $\phi_\omega(x) : \mathcal{X} \mapsto \mathbb{R}$ from $L_2(\Omega, \rho)$, such that $k(x, x') = \int_\Omega \phi_\omega(x)\, \phi_\omega(x')\, d\rho(\omega)$.*

Essentially, the integral representation relates the kernel function to a random process ω with measure $d\rho(\omega)$. Note that the integral representation may not be unique, *i.e.*, there may multiple operator $\mathcal{Q}$ for one kernel. For some kernels, we can construct the expectation explicitly, therefore, obtain a concrete $\mathcal{Q}$. Specifically, if the kernel is also continuous and shift invariant, *i.e.*, $k(x, x') = k(x - x')$ for $x \in \mathbb{R}^d$, then the integral representation specializes into a form characterized by inverse Fourier transformation (*e.g.*, [76, Theorem 6.6]),

Theorem 9 (Bochner) *A continuous, real-valued, symmetric and shift-invariant function $k(x - x')$ on $\mathbb{R}^d$ is a PD kernel if and only if there is a finite non-negative measure $\rho(\omega)$ on*

Table 3.1: Summary of kernels in [62, 17, 71, 72, 73, 74, 75] and their explicit features

Kernel	$k(x, x')$	$\phi_\omega(x)$	$\rho(\omega)$
Gaussian	$\exp(-\frac{\|x-x'\|_2^2}{2})$	$\exp(-i\omega^\top x)$	$(2\pi)^{-\frac{d}{2}} \exp(-\frac{\|\omega\|_2^2}{2})$
Laplacian	$\exp(-\|x - x'\|_1)$	$\exp(-i\omega^\top x)$	$\prod_{i=1}^d \frac{1}{\pi(1+\omega_i^2)}$
Cauchy	$\prod_{i=1}^d \frac{2}{1+(x_i-x_i')^2}$	$\exp(-i\omega^\top x)$	$\exp(-\|\omega\|_1)$
Matérn	$\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|x-x'\|_2}{\eta}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}\|x-x'\|_2}{\eta}\right)$	$\exp(-i\omega^\top x)$	$h(\nu, d, \eta)\left(\frac{2\nu}{\eta^2}+4\pi^2\|\omega\|_2^2\right)^{\nu+d/2}$
Dot Product	$\sum_{n=0}^\infty a_n \langle x, x'\rangle^n \quad a_n \geqslant 0$	$\sqrt{a_N p^{N+1}} \prod_{i=1}^N \omega_i^\top x$	$P[N = n] = \frac{1}{p^{n+1}}$
Polynomial	$(\langle x, x'\rangle + c)^p$	$\texttt{FFT}^{-1}(\texttt{FFT}(C_1 x) \odot \ldots \odot \texttt{FFT}(C_p x))$	$p(\omega_i^j \mid N = n) = \frac{1}{2}\frac{\omega_i^j+1}{2}\frac{1}{2}\frac{1-\omega_i^j}{2}$ $C_j = S_j D_j$ $D_j \in \mathbb{R}^{d\times d}\, S_j \in \mathbb{R}^{D\times d}$
Hellinger	$\sum_{i=1}^d \sqrt{x_i x_i'}$	$2\omega^\top \sqrt{x}$	$\frac{1}{2}\frac{\omega_i+1}{2}\frac{1}{2}\frac{1-\omega_i}{2},\ \omega_i \in \{-1, +1\}$
χ^2	$2\sum_{i=1}^d \frac{x_i x_i'}{x_i+x_i'}$	$\left[\exp(-i\omega \log x_j)\sqrt{x_j}\right]_{j=1}^d$	$sech(\pi\omega)$
Intersection	$\sum_{i=1}^d \min(x_i, x_i')$	$\left[\exp(-i\omega \log x_j)\sqrt{2x_j}\right]_{j=1}^d$	$\frac{1}{\pi(1+4\omega^2)}$
Jensen-Shannon	$\sum_{i=1}^d K_{JS}(x_i, x_i')$	$\left[\exp(-i\omega \log x_j)\sqrt{2x_j}\right]_{j=1}^d$	$\frac{sech(\pi\omega)}{\log 4(1+4\omega^2)}$
Skewed-χ^2	$2\prod_{i=1}^d \frac{\sqrt{x_i+c}\sqrt{x_i'+c}}{x_i+x_i'+2c}$	$\exp(-i\omega^\top \log(x + c))$	$\prod_{i=1}^d sech(\pi\omega_i)$
Skewed-Intersection	$\prod_{i=1}^d \min\left(\sqrt{\frac{x_i+c}{x_i'+c}}, \sqrt{\frac{x_i'+c}{x_i+c}}\right)$	$\exp(-i\omega^\top \log(x + c))$	$\prod_{i=1}^d \frac{1}{\pi(1+4\omega_i^2)}$
Exponential-Semigroup	$\exp(-\beta\sum_{i=1}^d \sqrt{x_i + x_j'})$	$\exp(-\omega^\top x)$	$\prod_{i=1}^d \frac{\beta}{2\sqrt{\pi}}\omega_i^{-\frac{3}{2}} \exp(-\frac{\beta}{4\omega_i})$
Reciprocal-Semigroup	$\prod_{i=1}^d \frac{\lambda}{x_i+x_i'+\lambda}$	$\exp(-\omega^\top x)$	$\prod_{i=1}^d \lambda \exp(-\lambda\omega_i)$
Arc-Cosine	$\frac{1}{\pi}\|x\|^n\|x'\|^n J_n(\theta)$	$(\omega^\top x)^n \max(0, \omega^\top x)$	$2\pi^{-\frac{d}{2}} \exp(-\frac{\|\omega\|_2^2}{2})$

D_j is random $\{\pm 1\}$ diagonal matrix and the columns of S_j are uniformly selected from $\{e_1, \ldots, e_D\}$. ν and η are positive parameters.

$h(\nu, d, \eta) = \frac{2^d \pi^{d/2}\Gamma(\nu+d/2)(2\nu)^\nu}{\Gamma(\nu)\eta^{2\nu}}$. K_ν is a modified Bessel function. $K_{JS}(x, x') = \frac{x}{2}\log_2 \frac{x+x'}{x} + \frac{x'}{2}\log_2 \frac{x+x'}{x'}$.

$\theta = \cos^{-1}\frac{x^\top x'}{\|x\|\|x'\|}$, $J_n(\theta) = (-1)^n (\sin\theta)^{2n+1}\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\right)^n\left(\frac{\pi-\theta}{\sin\theta}\right)$

$\mathbb{R}^d$, *such that*

$$k(x-x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x-x')}\, d\rho(\omega) = 2\int_{\mathbb{R}^d\times[0,2\pi]} \cos(\omega^\top x + b)\, \cos(\omega^\top x' + b)\, d\left(\rho(\omega)\times\rho(b)\right),$$

where $\rho(b)$ *is a uniform distribution on* $[0, 2\pi]$ *and* $\phi_\omega(x) = \sqrt{2}\cos(\omega^\top x + b)$.

For Gaussian RBF kernel, $k(x-x') = \exp(-\|x-x'\|^2/2\sigma^2)$, this yields a Gaussian distribution $\rho(\omega)$ with density proportional to $\exp(-\sigma^2\|\omega\|^2/2)$; for the Laplace kernel, this yields a Cauchy distribution; and for the Martern kernel, this yields the convolutions of the unit ball [16].

Similar representation where the explicit form of $\phi_\omega(x)$ and $\rho(\omega)$ are known can also be derived for rotation invariant kernel, $k(x, x') = k(\langle x, x'\rangle)$, using Fourier transformation on sphere [16]. For polynomial kernels, $k(x, x') = (\langle x, x'\rangle + c)^p$, a random tensor sketching approach can also be used [72]. Explicit random features have been designed for many other kernels, such as dot product kernel [71], additive/multiplicative class of homogeneous kernels [73], *e.g.*, Hellinger's, χ^2, Jensen-Shannon's and Intersection kernel, as well as kernels on Abelian semigroups [74]. We summarized these kernels with their explicit features and associated densities in Table 3.1.

Instead of finding the expectation decomposition for a given kernel to construct $\mathcal{Q}$, one can go the reverse direction, and construct kernels from the operator $\mathcal{Q}$, *i.e.*, random processes $d\rho\,(\omega)$ and basis functions $\phi_\omega\,(\cdot)$ (*e.g.*, [76]).

Theorem 10 *If* $k(x, x') = \int_\Omega \phi_\omega(x)^\top \phi_\omega(x')\, d\rho(\omega)$ *for a nonnegative measure* $\rho(\omega)$ *on* Ω *and* $\phi_\omega(x) : \mathcal{X} \mapsto \mathbb{R}^r$, *each component from* $L_2(\Omega, \rho)$, *then* $k(x, x')$ *is a PD kernel.*

For instance, we can define $\mathcal{Q}$ via Eq (2.1) by $\phi_\omega(x) = \cos(\omega^\top \psi_\theta(x) + b)$, where $\psi_\theta(x)$ can be a random convolution of the input x parametrized by θ, or $\phi_\omega(x) = [\phi_{\omega_1}(x), \phi_{\omega_2}(x), \ldots, \phi_{\omega_r}(x)]$, where $\phi_{\omega_1}(x)$ denote the random feature for kernel $k_1(x, x')$. The former $\mathcal{Q}$ with random features define a hierarchical kernel [75], and the latter one induce a linear combination of multiple kernels. It is worth to note that the Hellinger's, χ^2, Jensen-Shannon's and Intersection kernels in [73] are special cases of multiple kernels combination. For simplicity, we assume $\phi_w(x) \in \mathbb{R}$ following, and our algorithm is still applicable to $\phi_w(x) \in \mathbb{R}^r$.

In fact, Theorem 8 and Theorem 10 are just the concrete explanations to the equivalence between RKHS induced by $k(\cdot,\cdot)$ and the image of integral operator $\mathcal{Q}$ defined by random features $\phi_\omega(\cdot)$ and $\rho(\omega)$. For simplicity and convenience, in the following sections, we will derive the practical Algorithm 1 and 2 for optimization (2.4) from approximating the gradients w.r.t. the RKHS functions, *i.e.*, $f = \mathcal{Q}g$, with particular random feature representation, instead of the derivation based on functional gradient w.r.t. g. We emphasize that we can derive the *same algorithm* following the latter view too, while the theoretical analysis from the former view is relatively more clear.

3.3 Doubly Stochastic Functional Gradients

As we discussed in Chapter 2, by setting $\mathcal{T}$ as $\mathcal{Q}$ associated with a RKHS $\mathcal{H}$ and a PD kernel $k(x, x')$, denote $f = \mathcal{Q}g \in \mathcal{H}$, we can reformulate many kernel methods as finding a function $f^* \in \mathcal{H}$ via solving the optimization problem (2.4), *i.e.*,

$$\operatorname*{argmin}_{f\in\mathcal{H}} \quad R(f) := \mathbb{E}_{(x,y)}[\ell(f(x), y)] + \frac{\nu}{2}\|f\|_{\mathcal{H}}^2 \quad \Longleftrightarrow \quad \operatorname*{argmin}_{\|f\|_{\mathcal{H}}\leqslant B(\nu)} \quad \mathbb{E}_{(x,y)}[\ell(f(x), y)] \tag{3.2}$$

where $\nu > 0$ is a regularization parameter, $B(\nu)$ is a non-increasing function of ν, and the data (x, y) follow a distribution $p(x, y)$.

Stochastic functional gradient Given a data point $(x, y) \sim p(x, y)$ and $f \in \mathcal{H}$, following the definition of functional gradient in Section 2.1, the stochastic functional gradient of $\mathbb{E}_{(x,y)}[\ell(f(x), y)]$ with respect to $f \in \mathcal{H}$ is

$$\xi(\cdot) := \ell'(f(x), y)k(x, \cdot), \tag{3.3}$$

which is essentially a single data point approximation to the true functional gradient. Furthermore, we can represent the function $k(x, \cdot) = \mathcal{Q}g := \mathbb{E}_\omega[\phi_\omega(\cdot)\, g(\omega)]$ with $g(\omega) = \phi_\omega(x)$. Therefore, we can make an additional approximation to the stochastic functional gradient using a random feature $\phi_\omega(x)$ sampled according to $\rho(\omega)$. More specifically,

Doubly stochastic functional gradient Let $\omega \sim \rho(\omega)$, then the doubly stochastic gra-

dient of $\mathbb{E}_{(x,y)}[\ell(f(x), y)]$ with respect to $f \in \mathcal{H}$ is

$$\zeta(\cdot) := \ell'(f(x), y)\phi_\omega(x)\phi_\omega(\cdot). \tag{3.4}$$

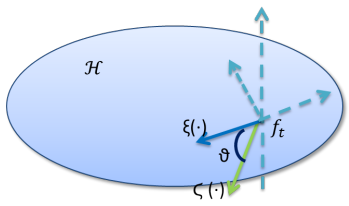


Figure 3.1: ϑ is the angle betwen $\xi(\cdot)$ and $\zeta(\cdot)$, $\zeta(\cdot)$ may be outside of $\mathcal{H}$.

Note that the stochastic functional gradient $\xi(\cdot)$ is in RKHS $\mathcal{H}$ but $\zeta(\cdot)$ may be outside $\mathcal{H}$, since $\phi_\omega(\cdot)$ may be outside the RKHS. For instance, for the Gaussian RBF kernel, the random feature $\phi_\omega(x) = \sqrt{2}\cos(\omega^\top x + b)$ can be outside of the RKHS associated with the kernel function.

These functional gradients are related by $\xi(\cdot) = \mathbb{E}_\omega[\zeta(\cdot)]$, which lead to *unbiased* estimators of the original functional gradient, *i.e.*,

$$\nabla R(f) = \mathbb{E}_{(x,y)}[\xi(\cdot)] + vf(\cdot), \tag{3.5}$$

$$\text{and} \quad \nabla R(f) = \mathbb{E}_{(x,y)}\mathbb{E}_\omega[\zeta(\cdot)] + vf(\cdot). \tag{3.6}$$

The unbiasedness of the gradient estimator Eq (3.5) relies on the fact we use the *same* $f \in \mathcal{H}$ to evaluate the $\ell'(f(x), y)$. Once we update the f with the doubly stochastic gradient, since we use Monte-Carlo approximation on random features through $\mathcal{Q}g$ to evaluate f, it will introduce extra error in f, the gradient estimator Eq (3.5) will be no longer unbiased due to the error introduced by calculating $\ell'(f(x), y)$. However, as we will prove later in Section 3.4, as long as the bias is comparable with the variance, the stochastic gradient descent still converges.

We emphasize that the source of randomness associated with the random feature is not present in the data, but artificially introduced by us. This is crucial for the development of our scalable algorithm in the next section. Meanwhile, it also creates additional challenges

in the analysis of the algorithm which we will deal with carefully.

3.3.1 Doubly Stochastic Kernel Machines

Algorithm 1 $\{\alpha_i\}_{i=1}^t = \text{Train}(p(x,y))$

Input: $\rho(\omega)$, $\phi_\omega(x)$, $\ell(f(x),y)$, ν.
for $i = 1,\ldots,t$ **do**
  Sample $(x_i, y_i) \sim p(x,y)$.
  Sample $\omega_i \sim \rho(\omega)$ with seed i.
  $f(x_i) = \textbf{Predict}(x_i, \{\alpha_j\}_{j=1}^{i-1})$.
  $\alpha_i = -\gamma_i \ell'(f(x_i), y_i)\phi_{\omega_i}(x_i)$.
  $\alpha_j = (1-\gamma_i\nu)\alpha_j$ for $j = 1,\ldots,i-1$.
end for

Algorithm 2 $f(x) = \text{Predict}(x, \{\alpha_i\}_{i=1}^t)$

Input: $\rho(\omega)$, $\phi_\omega(x)$.
Set $f(x) = 0$.
for $i = 1,\ldots,t$ **do**
  Sample $\omega_i \sim \rho(\omega)$ with seed i.
  $f(x) = f(x) + \alpha_i \phi_{\omega_i}(x)$.
end for

The first key intuition behind our algorithm originates from the property of stochastic gradient descent algorithm that as long as the bias of the stochastic gradient is controllable, *i.e.*, the bias and variance of the stochastic gradient are balanced, the convergence of the algorithm will still be guaranteed. In our algorithm, we will exploit this property and introduce *two* sources of randomness, one from data and another artificial, to scale up kernel methods.

The second key intuition behind our algorithm is that the random features used in the doubly stochastic functional gradients will be sampled according to *pseudo-random number generators*, where the sequences of apparently random samples can in fact be completely determined by an initial value (a seed). Although these random samples are not the "true" random sample in the purest sense of the word, however they suffice for our task in practice.

More specifically, our algorithm proceeds by making two stochastic approximation to the functional gradient in each iteration, and then descending using this noisy functional gradient. The overall algorithms for training and prediction is summarized in Algorith-

m 1. The training algorithm essentially just performs random feature sampling and doubly stochastic gradient evaluation, and maintains a collection of real number $\{\alpha_i\}$, which is computationally efficient and memory friendly. A crucial step in the algorithm is to sample the random features with "*seed i*". The seeds have to be aligned between training and prediction, and with the corresponding α_i obtained from each iteration. The learning rate γ_t in the algorithm needs to be chosen as $O(1/t)$, as shown by our later analysis to achieve the best rate of convergence. For now, we assume that we have access to the data generating distribution $p(x, y)$. This can be modified to sample uniformly randomly from a fixed dataset, without affecting the algorithm and the later convergence analysis. Let the sampled data and random feature parameters be $\mathcal{D}^t := \{(x_i, y_i)\}_{i=1}^t$ and $\boldsymbol{\omega}^t := \{\omega_i\}_{i=1}^t$ respectively after t iteration, the function obtained by Algorithm 1 is a simple additive form of the doubly stochastic functional gradients

$$f_{t+1}(\cdot) = f_t(\cdot) - \gamma_t(\zeta_t(\cdot) + \nu f_t(\cdot)) = \sum\nolimits_{i=1}^{t} a_t^i \zeta_i(\cdot), \quad \forall t > 1, \quad \text{and} \quad f_1(\cdot) = 0, \tag{3.7}$$

where $a_t^i = -\gamma_i \prod_{j=i+1}^{t}(1 - \gamma_j \nu)$ are deterministic values depending on the step sizes $\gamma_j (i \leqslant j \leqslant t)$ and regularization parameter ν. This simple form makes it easy for us to analyze its convergence.

Remark: We emphasize that the updates Eq (3.7) can also be derived directly from applying gradient descent to optimization (2.4), *i.e.*,

$$L(g) = \mathbb{E}_{x,y}\left[\ell\left(\mathbb{E}_{\omega\sim\rho(\omega)}\left[\phi_\omega(x)\, g(\omega)\right], y\right)\right] + \nu \|g\|_2^2 .$$

Particularly, we have $\nabla_g L = \mathbb{E}_{x,y}\left[\ell'\left(\mathbb{E}_{\omega\sim\rho(\omega)}\left[\phi_\omega(x)\, g(\omega)\right], y\right)\phi_\cdot(x)\right] + \nu g(\cdot)$, Then, by taking two stochastic approximation to $\mathbb{E}_{x,y}[\cdot]$ and the random feature ω in $\phi_\cdot(x)$, we can achieve the same updates as Eq (3.7). Indeed, from such perspective, the doubly stochastic gradient descent algorithm can be viewed as the infinite-dimension generalization of the stochastic coordinate descent algorithm.

We note that our algorithm can also take a mini-batch of points and random features at each step, and estimate an empirical covariance for preconditioning to achieve potentially

better performance.

Our algorithm is general and can be applied to most of the kernel machines which are formulated in the convex optimization (3.2) in a RKHS $\mathcal{H}$ associated with given kernel $k(x, x')$. We will instantiate the doubly stochastic gradients algorithms for a few commonly used kernel machines for different tasks and loss functions, *e.g.*, regression, classification, quantile regression, novelty detection and estimating divergence functionals/likelihood ratio. Interestingly, the Gaussian process regression, which is a Bayesian model, can also be reformulated as the solution to particular convex optimizations in RKHS, and therefore, be approximated by the proposed algorithm.

Kernel Support Vector Machine (SVM) Hinge loss is used in kernel SVM where $\ell(u, y) = \max\{0, 1 - uy\}$ with $y \in \{-1, 1\}$. We have $\ell'(u, y) = \begin{cases} 0 & \text{if } yu \geqslant 1 \\ -y & \text{if } yu < 1 \end{cases}$ and the step 5 in Algorithm 1 becomes

$$\alpha_i = \begin{cases} 0 & \text{if } y_i f(x_i) \geqslant 1 \\ \gamma_i y_i \phi_{\omega_i}(x_i) & \text{if } y_i f(x_i) < 1 \end{cases}.$$

Remark: [77] used squared hinge loss, $\ell(u, y) = \frac{1}{2}\max\{0, 1 - uy\}^2$, in ℓ_2-SVM. With this loss function, we have $\ell'(u, y) = \begin{cases} 0 & \text{if } yu \geqslant 1 \\ u - y & \text{if } yu < 1 \end{cases}$ and the step 5 in Algorithm 1 becomes

$$\alpha_i = \begin{cases} 0 & \text{if } y_i f(x_i) \geqslant 1 \\ \gamma_i (y_i - f(x_i)) \phi_{\omega_i}(x_i) & \text{if } y_i f(x_i) < 1 \end{cases}.$$

Kernel Logistic Regression Log-loss is used in kernel logistic regression for binary classification where $\ell(u, y) = \log(1 + \exp(-yu))$ with $y \in \{-1, 1\}$. We have $\ell'(u, y) = -\frac{y \exp(-yu)}{1+\exp(-yu)}$ and the step 5 in Algorithm 1 becomes

$$\alpha_i = \gamma_i y_i \frac{\exp(-y_i f(x_i))}{1 + \exp(-y_i f(x_i))} \phi_{\omega_i}(x_i).$$

For the multi-class kernel logistic regression, the

$$\ell(u, y) = -\sum_{c=1}^{C} \delta_c(y) u_c + \log\left(\sum_{c=1}^{C} \exp(u_c)\right)$$

where C is the number of categories, $u \in \mathbb{R}^{C\times 1}$, $y \in \{1, \ldots, C\}$ and $\delta_c(y) = 1$ only if $y = c$, otherwise $\delta_c(y) = 0$. In such scenario, we denote $\mathbf{f}(x_i) = [f^1(x_i), \ldots, f^C(x_i)]$, and therefore, the corresponding $\boldsymbol{\alpha} = [\alpha^1, \ldots, \alpha^C]$. The update rule for $\boldsymbol{\alpha}$ in Algorithm 1 is

$$\begin{aligned} \alpha_i^c &= \gamma_i\left(\delta_c(y_i) - \frac{\exp(f^c(x_i))}{\sum_{c=1}^{C}\exp(f^c(x_i))}\right)\phi_{\omega_i}(x_i), \quad \forall c = 1, \ldots, C, \\ \alpha_j^c &= (1 - \gamma_i\nu)\alpha_j^c \quad , \forall j < i, \forall c = 1, \ldots, C. \end{aligned}$$

Kernel Ridge Regression Square loss is used in kernel ridge regression where $\ell(u, y) = \frac{1}{2}(u - y)^2$. We have $\ell'(u, y) = (u - y)$ and the step 5 in Algorithm 1 becomes

$$\alpha_i = -\gamma_i(f(x_i) - y_i)\phi_{\omega_i}(x_i).$$

Kernel Robust Regression Huber's loss is used for robust regression [32] where

$$\ell(u, y) = \begin{cases} \frac{1}{2}(u - y)^2 & \text{if } |u - y| \leqslant 1 \\ |u - y| - \frac{1}{2} & \text{if } |u - y| > 1 \end{cases}.$$

We have $\ell'(u, y) = \begin{cases} (u - y) & \text{if } |u - y| \leqslant 1 \\ \operatorname{sign}(u - y) & \text{if } |u - y| > 1 \end{cases}$ and the step 5 in Algorithm 1 becomes

$$\alpha_i = \begin{cases} -\gamma_i(f(x_i) - y_i)\phi_{\omega_i}(x_i) & \text{if } |f(x_i) - y_i| \leqslant 1 \\ -\gamma_i \operatorname{sign}(f(x_i) - y_i)\phi_{\omega_i}(x_i) & \text{if } |f(x_i) - y_i| > 1 \end{cases}$$

Kernel Support Vector Regression (SVR) ϵ-insensitive loss function is used in kernel SVR where $\ell(u, y) = \max\{0, |u - y| - \epsilon\}$. We have

$$\ell'(u, y) = \begin{cases} 0 & \text{if } |u - y| \leqslant \epsilon \\ \text{sign}(u - y) & \text{if } |u - y| > \epsilon \end{cases},$$

and the step 5 in Algorithm 1 becomes

$$\alpha_i = \begin{cases} 0 & \text{if } |f(x_i) - y_i| \leqslant \epsilon \\ -\gamma_i \,\text{sign}(f(x_i) - y_i)\phi_{\omega_i}(x_i) & \text{if } |f(x_i) - y_i| > \epsilon \end{cases}.$$

Remark: Note that if we set $\epsilon = 0$, the ϵ-insensitive loss function will become absolute deviation, *i.e.*, $\ell(u, y) = |u - y|$. Therefore, we have the updates for **kernel least absolute deviation regression**.

Kernel Quantile Regression The loss function for quantile regression is $\ell(u, y) = \max\{\tau(y - u), (1 - \tau)(u - y)\}$. We have $\ell'(u, y) = \begin{cases} 1 - \tau & \text{if } u \geqslant y \\ -\tau & \text{if } u < y \end{cases}$ and the step 5 in Algorithm 1 becomes

$$\alpha_i = \begin{cases} \gamma_i(\tau - 1)\phi_{\omega_i}(x_i) & \text{if } f(x_i) \geqslant y_i \\ \gamma_i \tau \phi_{\omega_i}(x_i) & \text{if } f(x_i) < y_i \end{cases}.$$

Kernel Novelty Detection The loss function $\ell(u, \tau) = \max\{0, \tau - u\}$ [30] is proposed for novelty detection. Since τ is also a variable which needs to be optimized, the optimization problem is formulated as

$$\min_{\tau \in \mathbb{R}, f \in \mathcal{H}} \mathbb{E}_x[\ell(f(x), \tau)] + \frac{\nu}{2}\|f\|_{\mathcal{H}}^2 - \nu\tau,$$

and the gradient of $\ell(u,\tau)$ is

$$\frac{\partial \ell(u,\tau)}{\partial u} = \begin{cases} 0 & \text{if } u \geqslant \tau \\ -1 & \text{if } u < \tau \end{cases}, \quad \frac{\partial \ell(u,\tau)}{\partial \tau} = \begin{cases} 0 & \text{if } u \geqslant \tau \\ 1 & \text{if } u < \tau \end{cases}.$$

The step 5 in Algorithm 1 becomes

$$\alpha_i = \begin{cases} 0 & \text{if } f(x_i) \geqslant \tau_{i-1} \\ \gamma_i \phi_{\omega_i}(x_i) & \text{if } f(x_i) < \tau_{i-1} \end{cases}, \quad \tau_i = \begin{cases} \tau_{i-1} + \gamma_i \nu & \text{if } f(x_i) \geqslant \tau_{i-1} \\ \tau_{i-1} - \gamma_i (1-\nu) & \text{if } f(x_i) < \tau_{i-1} \end{cases}.$$

Kernel Density Ratio Estimation Based on the variational form of Ali-Silvey divergence, *i.e.*, $\mathbb{E}_p\big[r(\frac{q}{p})\big]$, where $r : \mathbb{R}^+ \to \mathbb{R}$ is a convex function with $r(1) = 0$, [31] proposed a nonparametric estimator for the logarithm of the density ratio, $\log \frac{q}{p}$, which is the solution of following convex optimization,

$$\min_{f \in \mathcal{H}} \ \mathbb{E}_q[\exp(f)] + \mathbb{E}_p[r^*(-\exp(f))] + \frac{\nu}{2}\|f\|_{\mathcal{H}}^2, \tag{3.8}$$

where r^* denotes the Legendre-Fenchel dual of r, $r(\tau) := \sup_\chi \chi\tau - r^*(\chi)$. In Kullback-Leibler (KL) divergence, the $r_{KL}(\tau) = -\log(\tau)$. Its Legendre-Fenchel dual is

$$r_{KL}^*(\tau) = \begin{cases} \infty & \text{if } \tau \geqslant 0 \\ -1 - \log(-\tau) & \text{if } \tau < 0 \end{cases}$$

Specifically, the optimization becomes

$$\begin{aligned} \min_{f \in \mathcal{H}} \ R(f) &= \mathbb{E}_{y\sim q}[\exp(f(y))] - \mathbb{E}_{x\sim p}[f(x)] + \frac{\nu}{2}\|f\|_{\mathcal{H}}^2 \\ &= 2\mathbb{E}_{z,x,y}\Big[\delta_1(z)\exp(f(y)) - \delta_0(z) f(x)\Big] + \frac{\nu}{2}\|f\|_{\mathcal{H}}^2. \end{aligned}$$

where $z \sim$ Bernoulli(0.5). Denote $\ell(u_x, u_y, z) = \delta_1(z)\exp(u_y) - \delta_0(z)u_x$, we have

$$\ell'(u_x, u_y, z) = \delta_1(z)\exp(u_y) - \delta_0(z)$$

and the the step 5 in Algorithm 1 becomes

$$\alpha_i = -2\gamma_i(\delta_1(z_i)\exp(f(y_i))\phi_{\omega_i}(y_i) - \delta_0(z_i)\phi_{\omega_i}(x_i)), \quad z_i \sim \text{Bernoulli}(0.5).$$

In particular, the x_i and y_i are not sampled in pair, they are sampled independently from $p(x)$ and $q(x)$ respectively.

[31] proposed another convex optimization based on $r_{KL}(\tau)$ whose solution is a nonparametric estimator for the density ratio. [78] designed $r_{nv}(\tau) = \max(0, \rho - \log\tau)$ for novelty detection. Similarly, the doubly stochastic gradients algorithm is also applicable to these loss functions.

Gaussian Process Regression The doubly stochastic gradients can be used for approximating the posterior of Gaussian process regression by reformulating the mean and variance of the predictive distribution as the solutions to the convex optimizations with particular loss functions. Let $y = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(x) \sim \mathcal{GP}(0, k(x, x'))$, given the dataset $\{x_i, y_i\}_{i=1}^n$, the posterior distribution of the function at the test point x_* can be derived as

$$f^*|X, \mathbf{y}, x^* \sim \mathcal{N}\left(k^{*\top}\left(K + \sigma^2 I\right)^{-1}\mathbf{y},\ k(x^*, x^*) - k^{*\top}\left(K + \sigma^2 I\right)^{-1}k^*\right) \tag{3.9}$$

where $K \in \mathbb{R}^{n\times n}$, $K_{ij} = K(x_i, x_j)$, $k^* = [k(x^*, x_1), \ldots, k(x^*, x_n)]^\top$ and $I \in \mathbb{R}^{n\times n}$ is the identity matrix.

Obviously, the posterior mean of the Gaussian process for regression can be thought as the solution to optimization problem (3.2) with square loss and setting $\nu = 2\sigma^2$. Therefore, the update rule for approximating the posterior mean will be the same as kernel ridge regression.

To compute the predictive variance, we need to evaluate the $k^{*\top}\left(K + \sigma^2 I\right)^{-1}k^*$. Following, we will introduce two different optimizations whose solutions can be used for evaluating the quantity.

1. Denote $\phi = [k(x_1, \cdot), \ldots, k(x_n, \cdot)]$, then

$$\begin{aligned} k^{*\top} \left(K + \sigma^2 I\right)^{-1} k^* &= k(x^*, \cdot)^\top \phi \left(\phi^\top \phi + \sigma^2 I\right)^{-1} \phi^\top k(x^*, \cdot) \\ &= k(x^*, \cdot)^\top \phi\phi^\top \left(\phi\phi^\top + \sigma^2 I\right)^{-1} k(x^*, \cdot) \end{aligned}$$

where the second equation based on identity $\left(\phi\phi^\top + \sigma^2 I\right)\phi = \phi\left(\phi^\top\phi + \sigma^2 I\right)$. Therefore, we just need to estimate the operator:

$$\Sigma = \mathcal{C}\left(\mathcal{C} + \frac{\sigma^2}{n} I\right)^{-1} \quad \text{where} \quad \mathcal{C} = \frac{1}{n}\phi\phi^\top = \frac{1}{n}\sum_{i=1}^{n} k(x_i, \cdot) \otimes k(x_i, \cdot). \tag{3.10}$$

We can express Σ as the solution to the following convex optimization problem

$$\min_{\Sigma} R(\Sigma) = \frac{1}{2n}\sum_{i=1}^{n} \|k(x_i, \cdot) - \Sigma k(x_i, \cdot)\|_{\mathcal{H}}^2 + \frac{\sigma^2}{2n}\|\Sigma\|_{HS}^2$$

where $\|\cdot\|_{HS}$ is the Hilbert-Schmidt norm of the operator. We can achieve the optimum by $\nabla R = 0$, which is equivalent to Eq (3.10).

Based on this optimization, we approximate the Σ_t using $\sum_{i\leqslant j, i=1}^{t} \theta_{ij}\phi_{\omega_i}(\cdot) \otimes \phi_{\omega_j}(\cdot)$ by doubly stochastic functional gradients. The update rule for θ is

$$\begin{aligned} \theta_{ij} &= \left(1 - \frac{\sigma^2}{n}\gamma_t\right)\theta_{ij}, \ \forall i \leqslant j < t \\ \theta_{it} &= -\gamma_t \sum_{j\geqslant i}^{t-1} \theta_{ij}\phi_{\omega_j'}(x_t)\phi_{\omega_t'}(x_t), \ \forall i < t \\ \theta_{tt} &= \gamma_t \phi_{\omega_t}(x_t)\phi_{\omega_t'}(x_t). \end{aligned}$$

Please refer to Appendix A.4 for the details of the derivation.

2. Assume that the testing points, $\{x_i^*\}_{i=1}^m$, are given beforehand, instead of approximating the operator Σ, we target on functions $F^* = [f_1^*, \ldots, f_m^*]^\top$ where $f_i^*(\cdot) =$

$$k(\cdot)^\top \left(K + \sigma^2 I\right)^{-1} k_i^*,$$

$$k(\cdot) = [k(x_1, \cdot), \ldots, k(x_2, \cdot)],$$
$$k_i^* = [k(x_i^*, x_1), \ldots, k(x_i^*, x_n)]^\top.$$

Estimating $f_i^*(\cdot)$ can be accomplished by solving the optimization problem (3.2) with square loss and setting $y_j = k(x_i^*, x_j), \forall j = 1, \ldots, n$, $\nu = 2\sigma^2$, leading to the same update rule as kernel ridge regression.

After we obtain these estimators, we can calculate the predictive variance on x_i^* by either $k(x_i^*, x_i^*) - \sigma(x_i^*, x_i^*)$ or $k(x_i^*, x_i^*) - f_i^*(x_i^*)$. We conduct experiments to justify the novel formulations for approximating both the mean and variance of posterior of Gaussian processes for regression, and the doubly stochastic update rule in Section.(3.6).

Note that, to approximate the operator Σ, doubly stochastic gradient requires $O(t^2)$ memory. Although we do not need to save the whole training dataset, which saves $O(dt)$ memory cost, this is still computationally expensive. When the m testing data are given, we estimate m functions and each of them requires $O(t)$ memory cost, the total cost will be $O(tm)$ by the second algorithm.

3.4 Theoretical Analysis

In this section, we will show that, both in expectation and with high probability, our algorithm can estimate the optimal function in the RKHS with rate $O(1/t)$, and achieve a generalization bound of $O(1/\sqrt{t})$. The analysis for our algorithm relies on the explicit expectation representation of the $\mathcal{Q}g$ with random features and has a new twist compared to previous analysis of stochastic gradient descent algorithms, since the random feature approximation results in an estimator which is outside the RKHS. Besides the analysis for stochastic functional gradient descent, we need to use martingales and the corresponding concentration inequalities to prove that the sequence of estimators, f_{t+1}, outside the RKHS converge to the optimal function, f^*, in the RKHS. We make the following standard assumptions ahead for later references:

Assumption 1 *There exists an optimal solution, denoted as f^*, to the problem of our interest (3.2).*

Assumption 2 *Loss function $\ell(u, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and its first-order derivative is L-Lipschitz continous in terms of the first argument.*

Assumption 3 *For any data $\{(x_i, y_i)\}_{i=1}^t$ and any trajectory $\{f_i(\cdot)\}_{i=1}^t$, there exists $M > 0$, such that $|\ell'(f_i(x_i), y_i)| \leqslant M$.*

Note in our situation M exists and $M < \infty$ since we assume bounded domain and the functions f_t we generate are always bounded as well.

Assumption 4 *There exists $\kappa > 0$ and $\phi > 0$, such that $k(x, x') \leqslant \kappa$, $|\phi_\omega(x)\phi_\omega(x')| \leqslant \phi, \forall x, x' \in \mathcal{X}, \omega \in \Omega$.*

For example, when $k(\cdot, \cdot)$ is the Gaussian RBF kernel, we have $\kappa = 1$, $\phi = 2$.

We now present our main theorems as below. We will only provide a short sketch of proofs here. The full proofs for the these theorems are given in the Appendix A.1-A.3.

Theorem 11 (Convergence in expectation) *When $\gamma_t = \frac{\theta}{t}$ with $\theta > 0$ such that $\theta\nu \in (1, 2) \cup \mathbb{Z}_+$,*

$$\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}\left[|f_{t+1}(x) - f^*(x)|^2\right] \leqslant \frac{2C^2 + 2\kappa Q_1^2}{t}, \quad \textit{for any } x \in \mathcal{X}$$

where $Q_1 = \max\left\{\|f^*\|_{\mathcal{H}}, (Q_0 + \sqrt{Q_0^2 + (2\theta\nu - 1)(1 + \theta\nu)^2\theta^2\kappa M^2})/(2\nu\theta - 1)\right\}$, with $Q_0 = 2\sqrt{2}\kappa^{1/2}(\kappa + \phi)LM\theta^2$, and $C^2 = 4(\kappa + \phi)^2M^2\theta^2$.

Theorem 12 (Convergence with high probability) *When $\gamma_t = \frac{\theta}{t}$ with $\theta > 0$ such that $\theta\nu \in \mathbb{Z}_+$ and $t \geqslant \theta\nu$, for any $x \in \mathcal{X}$, we have with probability at least $1 - 3\delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,*

$$|f_{t+1}(x) - f^*(x)|^2 \leqslant \frac{C^2 \ln(2/\delta)}{t} + \frac{2\kappa Q_2^2 \ln(2t/\delta) \ln^2(t)}{t},$$

where C is as above and $Q_2 = \max\left\{\|f^*\|_{\mathcal{H}}, Q_0 + \sqrt{Q_0^2 + \kappa M^2(1 + \theta\nu)^2(\theta^2 + 16\theta/\nu)}\right\}$, with $Q_0 = 4\sqrt{2}\kappa^{1/2}M\theta(8 + (\kappa + \phi)\theta L)$.

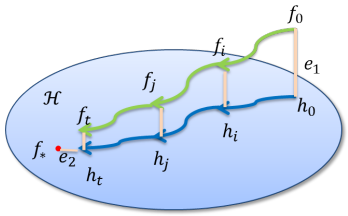


Figure 3.2: e_1 stands the error due to random features, and e_2 stands for the error due to random data.

Proof sketch: We focus on the convergence in expectation; the high probability bound can be established in a similar fashion. The main technical difficulty is that f_{t+1} may not be in the RKHS $\mathcal{H}$. The key of the proof is then to construct an intermediate function h_{t+1}, such that the difference between f_{t+1} and h_{t+1} and the difference between h_{t+1} and f^* can be bounded. More specifically,

$$h_{t+1}(\cdot) = h_t(\cdot) - \gamma_t(\xi_t(\cdot) + \nu h_t(\cdot)) = \sum\nolimits_{i=1}^{t} a_t^i \xi_i(\cdot), \quad \forall t > 1, \quad \text{and} \quad h_1(\cdot) = 0, \qquad (3.11)$$

where $\xi_t(\cdot) = \mathbb{E}_{\omega_t}[\zeta_t(\cdot)]$. Then for any x, the error can be decomposed as two terms

$$|f_{t+1}(x) - f^*(x)|^2 \leqslant 2 \underbrace{|f_{t+1}(x) - h_{t+1}(x)|^2}_{\text{error due to random features}} + 2\kappa \underbrace{\|h_{t+1} - f^*\|_{\mathcal{H}}^2}_{\text{error due to random data}}$$

For the error term due to random features, h_{t+1} is constructed such that $f_{t+1} - h_{t+1}$ is a martingale, and the stepsizes are chosen such that $|a_t^i| \leqslant \frac{\theta}{t}$, which allows us to bound the martingale. In other words, the choices of the stepsizes keep f_{t+1} close to the RKHS. For the error term due to random data, since $h_{t+1} \in \mathcal{H}$, we can now apply the standard arguments for stochastic approximation in the RKHS. Due to the additional randomness, the recursion is slightly more complicated, $e_{t+1} \leqslant \left(1 - \frac{2\nu\theta}{t}\right) e_t + \frac{\beta_1}{t}\sqrt{\frac{e_t}{t}} + \frac{\beta_2}{t^2}$, where $e_{t+1} = \mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}[\|h_{t+1} - f^*\|_{\mathcal{H}}^2]$, and β_1 and β_2 depends on the related parameters. Solving this recursion then leads to a bound for the second error term. ■

Theorem 13 (Generalization bound) *Let the true risk be* $R_{true}(f) = \mathbb{E}_{(x,y)}\left[\ell(f(x), y)\right]$.

Then with probability at least $1-3\delta$ *over* $(\mathcal{D}^t, \boldsymbol{\omega}^t)$*, and* C *and* Q_2 *defined as previously*

$$R_{true}(f_{t+1}) - R_{true}(f^*) \leqslant \frac{(C\sqrt{\ln(8\sqrt{e}t/\delta)} + \sqrt{2\kappa}Q_2\sqrt{\ln(2t/\delta)\ln(t)})L}{\sqrt{t}}.$$

Proof By the Lipschitz continuity of $\ell(\cdot, y)$ and Jensen's Inequality, we have

$$R_{true}(f_{t+1}) - R_{true}(f^*) \leqslant L\mathbb{E}_x|f_{t+1}(x) - f^*(x)| \leqslant L\sqrt{\mathbb{E}_x|f_{t+1}(x) - f^*(x)|^2} = L\|f_{t+1} - f^*\|_2.$$

Again, $\|f_{t+1} - f^*\|_2$ can be decomposed as two terms $O\left(\|f_{t+1} - h_{t+1}\|_2^2\right)$ and $O(\|h_{t+1} - f^*\|_{\mathcal{H}}^2)$, which can be bounded similarly as in Theorem 12 (see Corollary 36 in the appendix). ■

Remarks: The overall rate of convergence in expectation, which is $O(1/t)$, is indeed optimal. Classical complexity theory (see, e.g. reference in [70]) shows that to obtain ϵ-accuracy solution, the number of iterations needed for the stochastic approximation is $\Omega(1/\epsilon)$ for strongly convex case and $\Omega(1/\epsilon^2)$ for general convex case. Different from the classical setting of stochastic approximation, our case imposes not one but two sources of randomness/stochasticity in the gradient, which intuitively speaking, might require higher order number of iterations for general convex case. However, the variance of the random features only contributes *additively* to the constant in the final convergence rate. Therefore, our method is still able to achieve the same rate as in the classical setting. Notice that these bounds are achieved by adopting the classical stochastic gradient algorithm, and they may be further refined with more sophisticated techniques and analysis. For example, techniques for reducing variance of SGD proposed in [79], mini-batch and preconditioning [80, 81] can be used to reduce the constant factors in the bound significantly. Theorem 11 also reveals bounds in L_∞ and L_2 sense as in Appendix A.2. The choices of stepsizes γ_t and the tuning parameters given in these bounds are only for sufficient conditions and simple analysis; other choices can also lead to bounds in the same order.

3.5 Computation, Memory and Statistics Trade-off

To investigate the computation, memory and statistics trade-off, we will fix the desired L_2 error in the function estimation to ϵ, *i.e.*, $\|f - f^*\|_2^2 \leqslant \epsilon$, and work out the dependency of

other quantities on ϵ. These other quantities include the preprocessing time, the number of samples and random features (or rank), the number of iterations of each algorithm, and the computational cost and memory requirement for learning and prediction. We assume that the number of samples, n, needed to achieve the prescribed error ϵ is of the order $O(1/\epsilon)$, the same for all methods. Furthermore, we make no other regularity assumption about margin properties or the kernel matrix such as fast spectrum decay. Thus the required number of random feature (or ranks), r, will be of the order $O(n) = O(1/\epsilon)$ [58, 59, 64, 65].

We will pick a few representative algorithms for comparison, namely, **(i)** NORMA [68]: kernel methods trained with stochastic functional gradients; **(ii)** k-SDCA [67]: kernelized version of stochastic dual coordinate ascend; **(iii)** r-SDCA: first approximate the kernel function with random features, and then run stochastic dual coordinate ascend; **(iv)** n-SDCA: first approximate the kernel matrix using Nyström's method, and then run stochastic dual coordinate ascend; similarly we will combine Pegasos algorithm [82], stochastic block mirror descent (SBMD) [83], and random block coordinate descent (RBCD) [84] with random features and Nyström's method, and obtain **(v)** r-Pegasos, **(vi)** n-Pegasos, **(vii)** r-SBMD, **(viii)** n-SBMD, **(ix)** r-RBCD, and **(x)** n-RBCD, respectively. The comparisons are summarized below in Table. 3.2[1]

From Table 3.2, one can see that our method, r-SDCA, r-Pegasos, r-SBMD and r-RBCD achieve the best dependency on the dimension, d, of the data up to a log factor. However, often one is interested in increasing the number of random features as more data points are observed to obtain a better generalization ability, *e.g.*, *in streaming setting*. Then special procedures need to be designed for updating the r-SDCA, r-Pegasos, r-SBMD and r-RBCD solutions, which is not clear how to do easily and efficiently with theoretical guarantees. As a more refined comparison, our algorithm is also the cheapest in terms of per training iteration computation and memory requirement. We list the computational and memory requirements at a particular iteration $t < n$ for these five algorithms to achieve ϵ error in Table 3.3.

[1]We only considered general kernel algorithms in this section. For some specific loss functions, *e.g.*, hinge-loss, there are algorithms proposed to achieve better memory saving with extra training cost, such as support vector reduction technique [85].

Table 3.2: Comparison of computation and memory requirements of existing algorithms for kernel machines

Algorithms	Preprocessing Computation	Total Computation Cost		Total Memory Cost	
		Training	Prediction	Training	Prediction
Doubly SGD	$O(1)$	$O(d/\epsilon^2)$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
NORMA	$O(1)$	$O(d/\epsilon^2)$	$O(d/\epsilon)$	$O(d/\epsilon)$	$O(d/\epsilon)$
k-SDCA	$O(1)$	$O(d/\epsilon^2 \log(\frac{1}{\epsilon}))$	$O(d/\epsilon)$	$O(d/\epsilon)$	$O(d/\epsilon)$
r-SDCA	$O(1)$	$O(d/\epsilon^2 \log(\frac{1}{\epsilon}))$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
n-SDCA	$O(1/\epsilon^3)$	$O(d/\epsilon^2 \log(\frac{1}{\epsilon}))$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
r-Pegasos	$O(1)$	$O(d/\epsilon^2)$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
n-Pegasos	$O(1/\epsilon^3)$	$O(d/\epsilon^2)$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
r-SBMD	$O(1)$	$O(d/\epsilon^2)$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
n-SBMD	$O(1/\epsilon^3)$	$O(d/\epsilon^2)$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
r-RBCD	$O(1)$	$O(d/\epsilon^2 \log(\frac{1}{\epsilon}))$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$
n-RBCD	$O(1/\epsilon^3)$	$O(d/\epsilon^2 \log(\frac{1}{\epsilon}))$	$O(d/\epsilon)$	$O(1/\epsilon)$	$O(1/\epsilon)$

Table 3.3: Comparison of computation and memory requirements per iteration of existing algorithms for kernel machines, where b denotes the block size in algorithms SBMD and RBCD.

Algorithms	Computation per Iteration	Memory per Iteration	Iteration #
Doubly SGD	$\Theta(dt + t + t)$	$\Theta(t)$	$O(1/\epsilon)$
r-SDCA	$\Theta(dn + n + n)$	$\Theta(n)$	$O(1/\epsilon \log(\frac{1}{\epsilon}))$
r-Pegasos	$\Theta(dn + n + n)$	$\Theta(n)$	$O(1/\epsilon)$
r-SBMD	$\Theta(dn + n + n/b)$	$\Theta(n)$	$O(b/\epsilon)$
r-RBCD	$\Theta(dn^2 + n^2 + n/b)$	$\Theta(n)$	$O(\log(1/\epsilon))$

3.6 Experiments

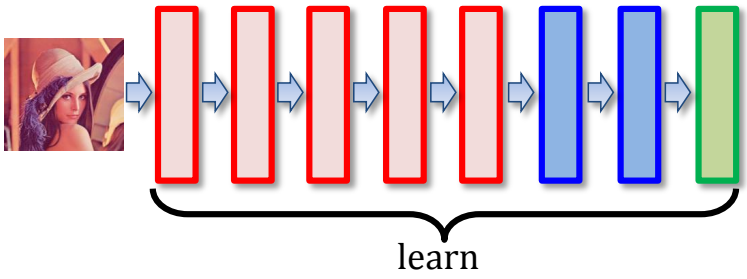
We show that our method compares favorably to other scalable kernel methods in medium scale datasets, and neural nets in large scale datasets. Below is a summary of the datasets used. A "yes" for the last column means that virtual examples (random cropping and mirror imaging of the original pictures) are generated for training. K-ridge stands for kernel ridge regression; GPR stands for Gaussian processes regression; K-SVM stands for kernel SVM; K-logistic stands for kernel logistic regression.

Experiment settings We first justify the doubly stochastic algorithm for Gaussian processes regression on dataset (1), comparing with NORMA. The dataset is medium size, so

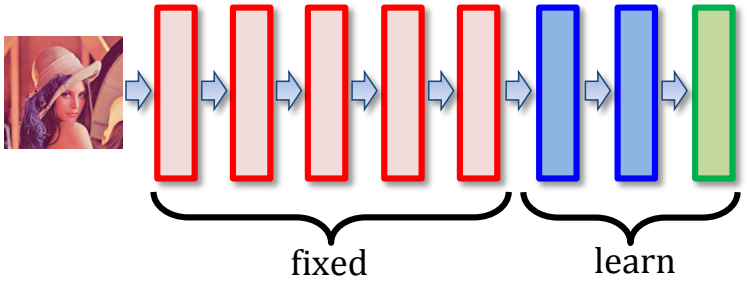
Table 3.4: Details of the tasks for evaluating doubly SGD

	Name	Model	# of samples	Input dim	Output range	Virtual
(1)	Synthetic	GPR	2^{11}	2	$[-1, 1.3]$	no
(2)	Synthetic	K-ridge	2^{20}	2	$[-1, 1.3]$	no
(3)	Adult	K-SVM	32K	123	$\{-1, 1\}$	no
(4)	MNIST 8M 8 vs. 6	K-SVM	1.6M	784	$\{-1, 1\}$	yes
(5)	Forest	K-SVM	0.5M	54	$\{-1, 1\}$	no
(6)	MNIST 8M	K-logistic	8M	1568	$\{0, \ldots, 9\}$	yes
(7)	CIFAR 10	K-logistic	60K	2304	$\{0, \ldots, 9\}$	yes
(8)	ImageNet	K-logistic	1.3M	9216	$\{0, \ldots, 999\}$	yes
(9)	QuantumMachine	K-ridge	6K	276	$[-800, -2000]$	yes
(10)	MolecularSpace	K-ridge	2.3M	2850	$[0, 13]$	no

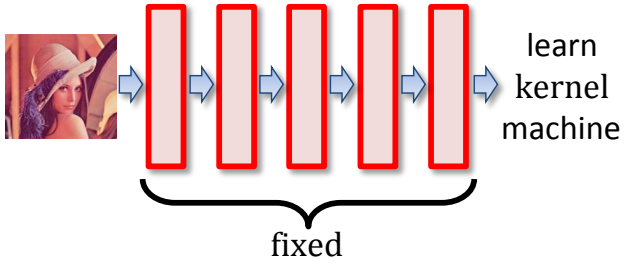
that the closed-form for posterior is tractable. For the large-scale datasets (2) — (5), we compare with the first seven algorithms for solving kernel methods discussed in Table 3.2. For the algorithms based on low rank kernel matrix approximation and random features, *i.e.*, pegasos and SDCA, we set the rank r or number of random features r to be 2^8. We use the same batch size for both our algorithms and the competitors. We adopted two stopping criteria for different purposes. We first stopped the algorithms when they pass through the entire dataset once (SC1). This stopping criterion is designed for justifying our motivation. By investigating the performances of these algorithms with different levels of random feature approximations but the same number of training samples, we could identify that the bottleneck of the performances of the vanilla methods with explicit feature will be their approximation ability. To further demonstrate the advantages of the proposed algorithm in computational cost, we also conduct experiments on datasets (3) – (5) running the competitors within the same time budget as the proposed algorithm (SC2). We do not count the preprocessing time of Nyström's method for n-Pegasos and n-SDCA, though it takes substantial amount of time. The algorithms are executed on the machine with AMD 16 2.4GHz Opteron CPUs and 200G memory. It should be noticed that this gives advantage to NORMA and k-SDCA which could save all the data in the memory. For fairness, we also record as many random features as the memory allowed. For datasets (6) — (8), we compare with neural nets for images ("jointly-trained"). In order to directly compare the performance of nonlinear classifiers rather than feature learning abilities, we also use the convolution layers of a trained neural net to extract features, then apply our algorithm and



(1) Jointly Trained Neural Nets



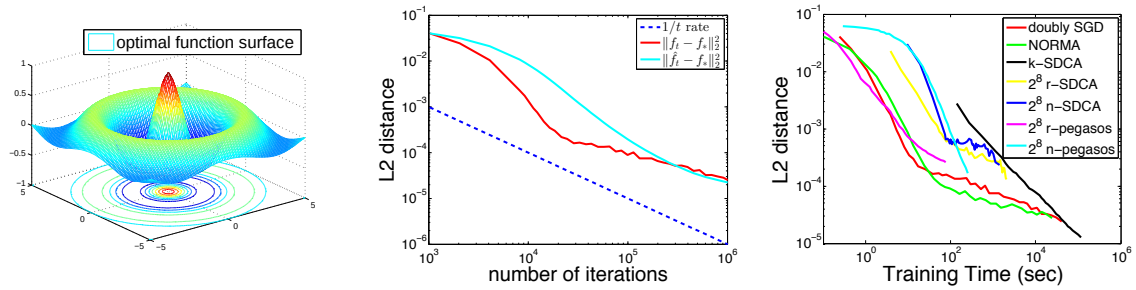
(2) Fixed Neural Nets



(3) Doubly Stochastic Kernel with Fixed Nets

Figure 3.3: Illustration of the neural nets structure in our experiments. The first several red layers are convolutions with max pooling layers. The following blue layers are fully connected layes. The green layer is the output layer which is multiclass logistic regression model.

a nonlinear neural net on top to learn classifiers ("fixed"). The structures of these neural nets in Figure 3.3. For datasets (9) and (10), we compare with the neural net described in [86] and use exactly the same input. In all the experiments, we select the batch size so that for each update, the computation resources can be utilized efficiently.



(1) 2D Synthetic Dataset (2) Convergence Rate (3) Accuracy vs. Time

Figure 3.4: Experimental results for kernel ridge regression on synthetic dataset.

3.6.1 Kernel Ridge Regression

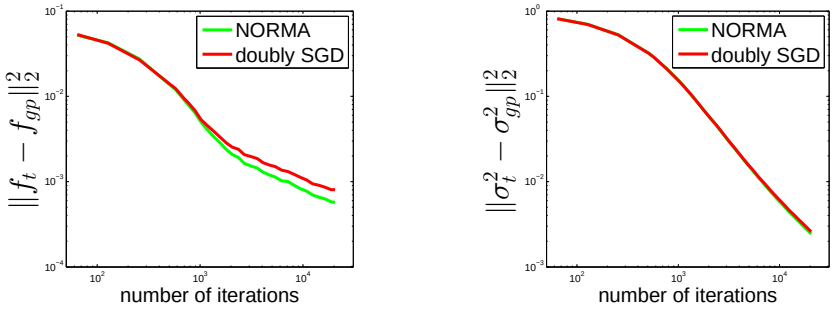
In this section, we compare our approach with alternative algorithms for kernel ridge regression on 2D synthetic dataset. The data are generated by

$$y = \cos(0.5\pi\|x\|_2)\exp(-0.1\pi\|x\|_2) + 0.1e$$

where $x \in [-5, 5]^2$ and $e \sim \mathcal{N}(0, 1)$. We use Gaussian RBF kernel with kernel bandwidth σ chosen to be 0.1 times the median of pairwise distances between data points (median trick). The regularization parameter ν is set to be 10^{-6}. The batch size and feature block are set to be 2^{10}.

The results are shown in Figure 3.4. In Figure 3.4(1), we plot the optimal functions generating the data. We justify our proof of the convergence rate in Figure 3.4(2). The blue dotted line is a convergence rate of $1/t$ as a guide. $\hat{f}_t$ denotes the average solution after t-iteration, *i.e.*, $\hat{f}_t(x) = \frac{1}{t}\sum_{i=1}^{t} f_i(x)$. It could be seen that our algorithm indeed converges in the rate of $O(1/t)$. In Figure 3.4 (3), we compare the first seven algorithms listed in the Table 3.2 for solving the kernel ridge regression.

The comparison on synthetic dataset demonstrates the advantages of our algorithm clearly. Our algorithm achieves comparable performance with NORMA, which uses full kernel, in similar time but less memory cost. The pegasos and SDCA using 2^8 random or Nyström features perform worse.



(1) Posterior Mean Convergence (2) Posterior Variance Convergence

Figure 3.5: Experimental results for Gaussian processes regression.

3.6.2 Gaussian Processes Regression

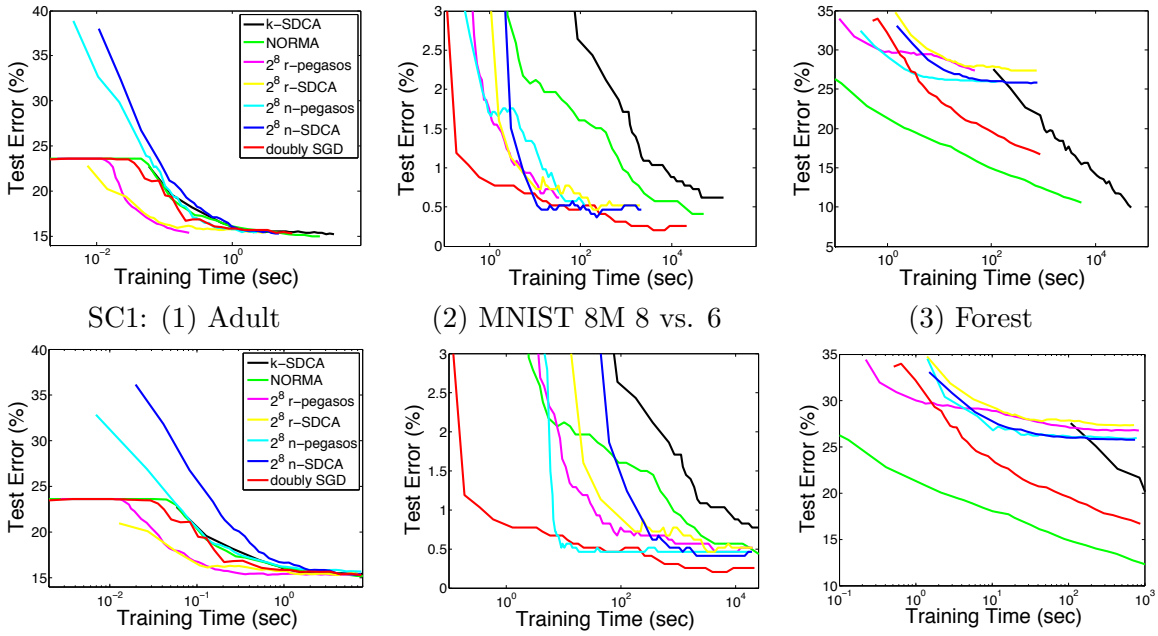
As we introduced in Section. (3.3.1), the mean and variance of posterior of Gaussian processes for regression problem can be formulated as solutions to some convex optimization problems. We conduct experiments on synthetic dataset for justification. Since the task is computing the posterior, we evaluate the performances by comparing the solutions to the posterior mean and variance, denoted as f_{gp} and σ_{gp}^2, obtained by closed-form (3.9). We select 2^{11} data from the same model in previous section for training and 2^{10} data for testing, so that the closed-form of posterior is tractable. We use Gaussian RBF kernel with kernel bandwidth σ chosen by median trick. The noise level σ^2 is set to be 0.1. The batch size is set to be 64 and feature block is set to be 512.

We compared the doubly stochastic algorithm with NORMA. The results are shown in Figure 3.5. Both the doubly stochastic algorithm and NORMA converge to the posterior, and our algorithm achieves comparable performance with NORMA in approximating both the mean and variance.

3.6.3 Kernel Support Vector Machine

We evaluate our algorithm solving kernel SVM on three datasets (3)–(5) comparing with other several algorithms listed in Table 3.2 using stopping criteria SC1 and SC2.

Adult We use Gaussian RBF kernel with kernel bandwidth obtained by median trick. The regularization parameter ν is set to be $1/(100n)$ where n is the number of training samples.



SC1: (1) Adult (2) MNIST 8M 8 vs. 6 (3) Forest

SC2: (4) Adult (5)MNIST 8M 8 vs. 6 (6) Forest.

Figure 3.6: Comparison with other kernel SVM solvers on datasets (3) – (5) with two different stopping criteria.

We set the batch size to be 2^6 and feature block to be 2^5. After going through the whole dataset one pass, the best error rate is achieved by NORMA and k-SDCA which is 15% while our algorithm achieves comparable result 15.3%. The performances are illustrated in Figure 3.6(1). Under the same time budget, all the algorithms perform similarly in Figure 3.6(4). The reason of flat region of r-pegasos, NORMA and the proposed method on this dataset is that Adult dataset is unbalanced. There are about 24% positive samples while 76% negative samples.

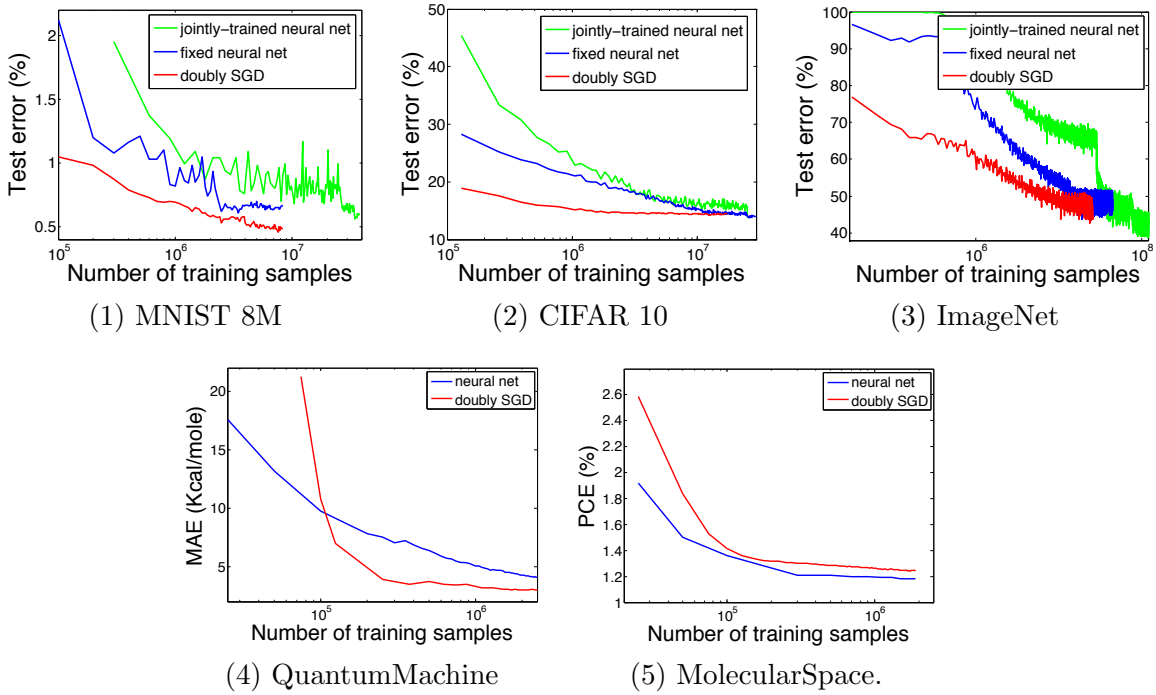
MNIST 8M 8 vs. 6 We first reduce the dimension to 50 by PCA and use Gaussian RBF kernel with kernel bandwidth $\sigma = 9.03$ obtained by median trick. The regularization parameter ν is set to be $1/n$ where n is the number of training samples. We set the batch size to be 2^{10} and feature block to be 2^8. The results are shown in Figure 3.6(2) and (5) under SC1 and SC2 respectively. Under both these two stopping criteria, our algorithm achieves the best test error 0.26% using similar training time.

Forest We use Gaussian RBF kernel with kernel bandwidth obtained by median trick. The regularization parameter ν is set to be $1/n$ where n is the number of training samples. We set the batch size to be 2^{10} and feature block to be 2^8. In Figure 3.6(3), we shows the performances of all algorithms using SC1. NORMA and k-SDCA achieve the best error rate, which is 10%, while our algorithm achieves around 15%, but still much better than the pegasos and SDCA with 2^8 features. In the same time budget, the proposed algorithm performs better than all the alternatives except NORMA in Figure 3.6(6).

As seen from the performance of pegasos and SDCA on Adult and MNIST, using fewer features does not deteriorate the classification error. This might be because there are cluster structures in these two binary classification datasets. Thus, they prefer low rank approximation rather than full kernel. Different from these two datasets, in the forest dataset, algorithms with full kernel, *i.e.*, NORMA and k-SDCA, achieve best performance, which indicates that there might not be low-rank structure in the kernel matrix. Therefore, the low-rank approximation obtained by the Nyström's method will lead to large approximation error. Indeed, [87] investigated the spectral structure of Gaussian kernel matrix on the forest dataset and justified our conclusion empirically. With more random features, our algorithm performs much better than pegasos and SDCA under both SC1 and SC2. Our algorithm is preferable for this scenario, *i.e.*, huge dataset with sophisticated decision boundary in the RKHS induced by high-rank kernels. Although utilizing full kernel could achieve better performance, the computation and memory requirement for the kernel on huge dataset are costly. To learn the sophisticated boundary in RKHS without low-rank kernel structure while still considering the computational and memory cost, we need to efficiently approximate the kernel in $O(\frac{1}{\epsilon})$ with $O(n)$ random features at least. Our algorithm could handle so many random features efficiently in both computation and memory cost, while for pegasos and SDCA such operation is prohibitive.

3.6.4 Classification Comparisons to Convolution Neural Networks

We also compare our algorithm with the state-of-the-art neural network. In these experiments, the block size is set to be $O(10^4)$. Compared to the number of samples, $O(10^8)$, this block size is reasonable.



(1) MNIST 8M (2) CIFAR 10 (3) ImageNet

(4) QuantumMachine (5) MolecularSpace.

Figure 3.7: Comparison with neural networks on datasets (6) – (10).

MNIST 8M In this experiment, we compare to a variant of LeNet-5 [88], where all tanh units are replaced with rectified linear units. We also use more convolution filters and a larger fully connected layer. Specifically, the first two convolutions layers have 16 and 32 filters, respectively, and the fully connected layer contains 128 neurons. We use kernel logistic regression for the task. We extract features from the last max-pooling layer with dimension 1568, and use Gaussian RBF kernel with kernel bandwidth σ equaling to four times the median pairwise distance. The regularization parameter ν is set to be 0.0005.

The result is shown in Figure 3.7(1). As expected, the neural net with pre-learned features is faster to train than the jointly-trained one. However, our method is much faster compared to both methods. In addition, it achieves a lower error rate (0.5%) compared to the 0.6% error provided by the neural nets.

CIFAR 10 In this experiment, we compare to a neural net with two convolution layers (after contrast normalization and max-pooling layers) and two local layers that achieves 11% test error on CIFAR 10 [89]. The features are extracted from the top max-pooling layer from a trained neural net with 2304 dimension. We use kernel logistic regression for

this problem. The kernel bandwidth σ for Gaussian RBF kernel is again four times the median pairwise distance. The regularization parameter ν is set to be 0.0005. We also perform a PCA (without centering) to reduce the dimension to 256 before feeding to our method.

The result is shown in Figure 3.7(2). The test error for our method drops significantly faster in the earlier phase, then gradually converges to that achieved by the neural nets. Our method is able to produce the same performance within a much restricted time budget.

ImageNet In this experiment, we compare our algorithm with the neural nets on the ImageNet 2012 dataset, which contains 1.3 million color images from 1000 classes. Each image is of size 256×256, and we randomly crop a 240×240 region with random horizontal flipping. The jointly-trained neural net is Alex-net [90]. The 9216 dimension features for our classifier and fixed neural net are from the last pooling layer of the jointly-trained neural net. The kernel bandwidth σ for Gaussian RBF kernel is again four times the median pairwise distance. The regularization parameter ν is set to be 0.0005.

Test error comparisons are shown in Figure 3.7(3). Our method achieves a test error of 44.5% by further max-voting of 10 transformations of the test set while the jointly-trained neural net arrives at 42% (without variations in color and illumination). At the same time, fixed neural net can only produce an error rate of 46% with max-voting. There may be some advantages to train the network jointly such that the layers work together to achieve a better objective. Although there is still a gap to the best performance by the jointly-trained neural net, our method comes very close with much faster convergence rate. Moreover, it achieves superior performance than the neural net trained with pre-learned features, both in accuracy and speed.

3.6.5 Regression Comparisons to Neural Networks

We test our algorithm for kernel ridge regression with neural network proposed in [86] on two large-scale real-world regression datasets, (9) and (10) in Table 3.4. To our best knowledge, this is the first comparison between kernel ridge regression and neural network on the dataset MolecularSpace.

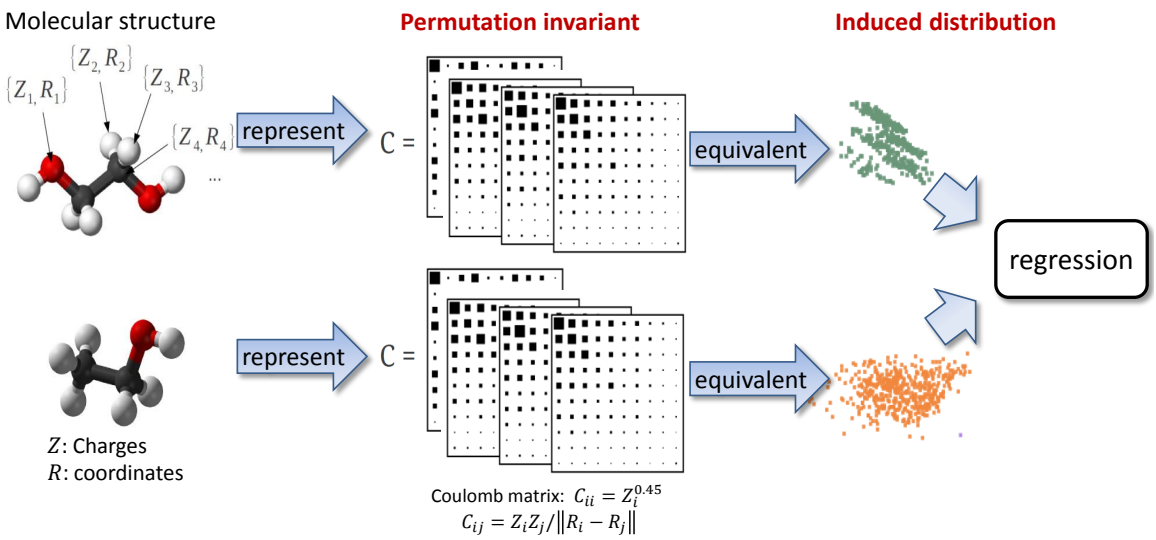


Figure 3.8: The computational procedure for predicting molecular property from molecular structure.

QuantumMachine In this experiment, we use the same binary representations converted based on random Coulomb matrices as in [86]. We first generate a set of randomly sorted coulomb matrices for each molecule. And then, we break each dimension of the Coulomb matrix apart into steps and convert them to the binary predicates. Predictions are made by taking average of all prediction made on various Coulomb matrices of the same molecule. The procedure is illustrated in Figure. 3.8. For this experiment, 40 sets of randomly permuted matrices are generated for each training example and 20 for each test example. We use Gaussian kernel with kernel bandwidth $\sigma = 60$ obtained by median trick. The batch size is set to be 50000 and the feature block is 2^{11}. The total dimension of random features is 2^{20}.

The results are shown in Figure 3.7(4). In QuantumMachine dataset, our method achieves Mean Absolute Error (MAE) of 2.97 kcal/mole, outperforming neural nets results, 3.51 kcal/mole. Note that this result is already close to the 1 kcal/mole required for chemical accuracy.

MolecularSpace In this experiment, the task is to predict the power conversion efficiency (PCE) of the molecule. This dataset of 2.3 million molecular motifs is obtained from the Clean Energy Project Database. We use the same feature representation as for "Quan-

tumMachine" dataset [86]. We set the kernel bandwidth of Gaussian RBF kernel to be 290 by median trick. The batch size is set to be 25000 and the feature block is 2^{11}. The total dimension of random features is 2^{20}.

The results are shown in Figure 3.7(5). It can be seen that our method is comparable with neural network on this 2.3 million dataset.

3.7 Summary

The integral operator view of RKHS functions provides us a new representation by which we can recast most of the kernel methods as the special case of the unified framework, *i.e.*, learning over functions problems.

From such a new perspective, we derive algorithm which avoids the explicit computation and storage of the kernel matrices, so that we make kernel methods scalable for large-scale datasets. Specifically, by introducing artificial randomness associated with kernels besides the random data samples, we propose doubly stochastic functional gradient for kernel machines which makes the kernel machines efficient in both computation and memory requirement. Our algorithm successfully reduces the memory requirement of kernel machines from $O(dn)$ to $O(n)$. Meanwhile, we also show that our algorithm achieves the optimal rate of convergence, $O(1/t)$, for strongly convex stochastic optimization. We compare our algorithm on both classification and regression problems with the state-of-the-art neural networks as well as some other competing algorithms for kernel methods on several large-scale datasets. With our efficient algorithm, kernel methods could perform comparable to sophisticated-designed neural network empirically.

The theoretical analysis, which provides the rate of convergence *independent* to the dimension, is also highly non-trivial. It twists martingale techniques and the vanilla analysis for stochastic gradient descent and provides a new tool for analyzing optimization in infinite-dimensional spaces, which could be of independent interest.

PART II: LEARNING OVER DISTRIBUTIONS

As we show in Section 2, when we instantiate the operator $\mathcal{T} := \langle \log p(x|\theta), \cdot \rangle$, $G(q)$ as KL-divergence regularization, and restrict the feasible domain as the density space, with the linear loss, the Bayesian inference can be obtained from Eq (2.3) as a special case, *i.e.*,

$$\min_{q \in \mathcal{P}} \; -\frac{1}{N} \sum_{i=1}^{N} \int q(\theta) \log p(x|\theta) \, d\theta + \frac{1}{N} \int q(\theta) \log \frac{q(\theta)}{p(\theta)} d\theta, \tag{3.12}$$

The above optimization can be understood as treating the inputs as $p(x|\theta)$ and searching for a density $q(\theta)$. Therefore, Bayesian inference can be recast as *learning over distributions* from the proposed integral operator framework.

Such a novel view replaces the intractable integral in Bayes' rule by an optimization, and thus, will lead to the *particle mirror descent* algorithm in Chapter 4, which is rooted in functional analysis, stochastic optimization and Monte-Carlo approximation. The proposed algorithm can tackle the notorious difficulty in Bayesian inference for the integral calculation when Bayes' rule does not result in a tractable closed-form, while most existing approximate inference algorithms lack either scalability or rigorous guarantees.

CHAPTER 4

PARTICLE MIRROR DESCENT IN DENSITY SPACE

Bayesian methods are attractive because of their ability in modeling complex data and capturing uncertainty in parameters. The crux of Bayesian inference is to compute the posterior distribution, $p(\theta|X) \propto p(\theta)\prod_{n=1}^{N} p(x_n|\theta)$, of a parameter $\theta \in \mathbb{R}^d$ given a set of N data points $X = \{x_n\}_{n=1}^{N}$ from $\mathbb{R}^D$, with a prior distribution $p(\theta)$ and a model of data likelihood $p(x|\theta)$. For many non-trivial models from real-world applications, the prior might not be conjugate to the likelihood or might contain hierarchical structure. Therefore, computing the posterior often results in intractable integration and poses computational challenges.

Recall that as introduced in Chapter 2, the posterior from the Bayes' rule can be rewritten as the solution to a special case of the unified framework (2.3), *i.e.*,

$$\min_{q\in\mathcal{P}} \; -\frac{1}{N}\sum_{i=1}^{N}\int q(\theta)\log p(x|\theta)\,d\theta + \frac{1}{N}\int q(\theta)\log\frac{q(\theta)}{p(\theta)}d\theta, \tag{4.1}$$

we can avoid the general intractable integral in Bayes' rule and target on solving the optimization. We propose *particle mirror descent* algorithm which exploits the stochastic functional mirror descent where one descends in the density space using a small batch of data points at each iteration, and particle filtering where one uses samples to approximate a function. We prove result that, with m particles, the proposed algorithm provides a posterior density estimator that converges in terms of KL-divergence to the true posterior in rate $O(1/\sqrt{m})$. We demonstrate competitive empirical performances of the proposed algorithm compared to several approximate inference algorithms in mixture models, logistic regression, sparse Gaussian processes and latent Dirichlet allocation on large-scale datasets.

4.1 Introduction

Two longstanding challenges in approximate Bayesian inference are i) *provable convergence* and ii) *data-intensive computation* at each iteration. MCMC is a general algorithm known to generate samples from distribution that converges to the true posterior. However, in order to generate a single sample at every iteration, it requires a complete scan of the dataset and evaluation of the likelihood at each data point, which is computationally expensive. To address this issue, approximate sampling algorithms have been proposed which use only a small batch of data points at each iteration [*e.g.* 91, 92, 93, 94]. [91, 92] extend the sequential Monte Carlo (SMC) to Bayesian inference on static models. However, these algorithms rely on Gaussian distribution or kernel density estimator as transition kernel for efficiency, which breaks down the convergence guarantee of SMC. On the other hand, the stochastic Langevin dynamics algorithm (SGLD) [93] and its derivatives [95, 96, 97] combine ideas from stochastic optimization and Hamiltonian Monte Carlo, and are proven to converge in terms of integral approximation, as recently shown in [98, 99]. Still, it is unclear whether the *dependent* samples generated reflects convergence to the true posterior. FireflyMC [94], which is a special case in pseudo-marginal Metropolis-Hastings (MH) family [100], introduces auxiliary variables to switch on and off data points to save computation for likelihood evaluations, but this algorithm requires the knowledge of lower bounds of likelihood that is model-specific and may be hard to calculate. The Austerity MCMC [101, 100] exploits the stochastic estimation of the acceptance ratio in the MH step with carefully designed statistical testing to balance the bias induced by sub-sampling. The algorithm performs well in practical for large-scale Bayesian inference, however, their theoretical analysis relies on strong assumptions on the distribution of the average log-likelihoods difference for designing the statistics, which might not generally hold.

In another line of research, the variational inference algorithms [102, 103, 104] attempt to approximate the entire posterior density by optimizing information divergence [105]. The recent derivatives [106] avoid examination of all the data in each update. However, the major issue for these algorithms is the absence of theoretical guarantees. This is due largely to the fact that variational inference algorithms typically choose a parametric family to

approximate the posterior density, which can be far from the true posterior, and require to solve a highly non-convex optimization problem. In most cases, these algorithms optimize over simple exponential family for tractability. More flexible variational families have been explored but largely restricted to mixture models [107, 108]. In these cases, it is often difficult to quantify the approximation and optimization error at each iteration, and analyze how the error accumulates across the iterations. Therefore, a provably convergent variational inference algorithm is still needed.

4.1.1 Contributions

In this chapter, we present such a simple and provable nonparametric inference algorithm, *particle mirror descent* (PMD), to iteratively approximate the posterior density. PMD relies on the connection that Bayes' rule can be expressed as the solution to a convex optimization problem over the density space [26, 1, 109]. However, directly solving the optimization will lead to both computational and representational issues: one scan over the entire dataset at each iteration is needed, and the exact function update has no closed-form. To address these issues, we draw inspiration from two sources: (**i**) stochastic mirror descent, where one can instead descend in the density space using a small batch of data points at each iteration; and (**ii**) particle filtering and kernel density estimation, where one can maintain a tractable approximate representation of the density using samples. In summary, PMD possesses a number of desiderata:

Simplicity PMD applies to many probabilistic models, even with *non-conjugate priors*. The algorithm is summarized in just a few lines of codes, and only requires the value of likelihood and prior, unlike other approximate inference techniques [93, 108, 110, 106, *e.g.*], which typically require their first and/or second-order derivatives.

Flexibility Different from other variational inference algorithms, which sacrifice the model flexibility for tractability, our method approximates the posterior by particles or kernel density estimator. The flexibility of nonparametric model enables PMD to capture multi-modal in posterior.

Stochasticity At iteration t, PMD only visits a mini-batch of data to compute the stochastic functional gradient, and samples $O(t)$ points from the solution. Hence, it avoids

scanning over the whole dataset in each update.

Theoretical guarantees We show the density estimator provided by PMD converges in terms of both integral approximation and KL-divergence to the true posterior density in rate $O(1/\sqrt{m})$ with m particles. To our best knowledge, these results are the first of the kind in Bayesian inference for estimating posterior.

In the remainder, we will introduce the optimization view of Bayes' rule before presenting our algorithm, and then we provide both theoretical and empirical supports of PMD.

Throughout this section, we denote KL as the Kullback-Leibler divergence, function $q(\theta)$ as q, a random sequence as $\theta_{[t]} := [\theta_1, \ldots, \theta_t]$, integral $f(\cdot)$ w.r.t. some measure $\mu(\theta)$ over support Ω as $\int f(\theta)\mu(d\theta)$, or $\int f(\theta)d\theta$ without ambiguity, $\langle \cdot, \cdot \rangle_{L_2}$ as the L_2 inner product, and $\| \cdot \|_p$ as the L_p norm for $1 \leqslant p \leqslant \infty$.

4.2 Optimization View of Bayesian Inference

Our algorithm stems from the connection between Bayes' rule and the optimization framework (2.3). [26, 1, 109] showed that Bayes' rule

$$p(\theta|X) = \frac{p(\theta)\prod_{n=1}^{N} p(x_n|\theta)}{p(X)}$$

where $p(X) = \int p(\theta)\prod_{n=1}^{N} p(x_n|\theta)d\theta$, can be obtained by solving the speical case of the unified integral operator framework (2.3),

$$\min_{q(\theta)\in\mathcal{P}} L(q) := -\sum_{n=1}^{N}\left[\int q(\theta)\log p(x_n|\theta)\,d\theta\right] + KL(q(\theta)\,||\,p(\theta)), \tag{4.2}$$

where $\mathcal{P}$ is the valid density space. The objective, $L(q)$, is continuously differentiable with respect to $q \in \mathcal{P}$ and one can further show that

Lemma 14 *Objective function $L(q)$ defined on $q(\theta) \in \mathcal{P}$ is 1-strongly convex w.r.t. KL-divergence.*

Despite of the closed-form representation of the optimal solution, it can be challenging to compactly represent, tractably compute, or efficiently sample from the solution. The

normalization, $p(X) = \int p(\theta) \prod_{n=1}^{N} p(x_n|\theta) d\theta$, involves high dimensional integral and typically does not admit tractable closed-form computation. Meanwhile, the product in the numerator could be arbitrarily complicated, making it difficult to represent and sample from. However, such an optimization with integral operator perspective provides us a way to tackle these challenges by leveraging recent advances from optimization algorithms.

4.2.1 Stochastic Mirror Descent in Density Space

We will resort to stochastic optimization to avoid scanning the entire dataset for each gradient evaluation. The stochastic mirror descent [70] expands the usual stochastic gradient descent scheme to problems with non-Euclidean geometries, by applying unbiased stochastic subgradients and Bregman distances as prox-map functions. We now explain in details, the stochastic mirror descent algorithm in the context of Bayesian inference.

At t-th iteration, given a data point x_t drawn randomly from the dataset, the stochastic functional gradient of $L(q)$ with respect to $q(\theta) \in L_2$ is $g_t(\theta) = \log(q(\theta)) - \log(p(\theta)) - N \log p(x_t|\theta)$. The stochastic mirror descent iterates over the prox-mapping step $q_{t+1} = \mathbf{P}_{q_t}(\gamma_t g_t)$, where $\gamma_t > 0$ is the stepsize and

$$\mathbf{P}_q(g) := \text{argmin}_{\widehat{q}(\theta) \in \mathcal{P}} \{ \langle \widehat{q}, g \rangle_{L_2} + KL(\widehat{q} \| q) \}.$$

Since the domain is density space, KL-divergence is a natural choice for the prox-function. The prox-mapping therefore admits the closed-form

$$\begin{aligned} q_{t+1}(\theta) &= q_t(\theta) \exp(-\gamma_t g_t(\theta))/Z \\ &= q_t(\theta)^{1-\gamma_t} p(\theta)^{\gamma_t} p(x_t|\theta)^{N\gamma_t}/Z, \end{aligned} \tag{4.3}$$

where $Z := \int q_t(\theta) \exp(-\gamma_t g_t(\theta))\, d\theta$ is the normalization. This update is similar the Bayes' rule. However, an important difference here is that the posterior is updated using the *fractional power* of the previous solution, the prior and the likelihood. Still computing $q_{t+1}(\theta)$ can be intractable due to the normalization Z.

4.2.2 Error Tolerant Stochastic Mirror Descent

To handle the intractable integral normalization at each prox-mapping step, we will consider a modified version of the stochastic mirror descent algorithm which can tolerate additional error in the prox-mapping step. Given $\epsilon \geqslant 0$ and $g \in L_2$, we define the ϵ-prox-mapping of q as the set

$$\mathbf{P}_q^{\epsilon}(g) := \left\{\widehat{q} \in \mathcal{P} \,:\, \langle g + \log \widehat{q} - \log q, \widehat{q} - q' \rangle_{L_2} \leqslant \epsilon, \forall q' \in \mathcal{P}\right\} \tag{4.4}$$

and consider the update $\tilde{q}_{t+1}(\theta) \in \mathbf{P}_{\tilde{q}_t}^{\epsilon_t}(\gamma_t g_t)$. When $\epsilon_t = 0, \forall t$, this reduces to the usual stochastic mirror descent algorithm. The classical results regarding the convergence rate can also be extended as below

Theorem 15 *Let $q^* = \operatorname{argmin}_{q \in \mathcal{P}} L(q)$, stochastic mirror descent with inexact prox-mapping after T steps gives the recurrence:*

$$\mathbb{E}[KL(q^* || \tilde{q}_{t+1})] \leqslant \epsilon_t + (1 - \gamma_t)\mathbb{E}[KL(q^* || \tilde{q}_t)] + \frac{\gamma_t^2}{2}\mathbb{E}\|g_t\|_{\infty}^2, \quad \forall t \leqslant T.$$

Remark: As shown in the classical analysis of stochastic mirror descent, we could also provide a non-asymptotic convergence results in terms of objective error at average solutions, *e.g.*, simple average $\bar{q}_T = \sum_{t=1}^T \gamma_t \tilde{q}_t / \sum_{t=1}^T \gamma_t$ in Appendix B.2.

Remark: For simplicity, we present the algorithm with stochastic gradient estimated by a single data point. The mini-batch trick is also applicable to reduce the variance of stochastic gradient, and convergence remains the same order but with an improved constant.

Allowing error in each step gives us room to design more flexible algorithms. Essentially, this implies that we can *approximate* the intermediate density by some *tractable representation*. As long as the approximation error is not too large, the algorithm will still converge; and if the approximation does not involve costly computation, the overall algorithm will still be efficient. Indeed, we propose two approximations as the solution to the ϵ-prox-mapping $\mathbf{P}_q^{\epsilon}(g)$ in following section.

4.3 Particle Mirror Descent Algorithm

We introduce two efficient approximations to Eq (4.3), which performs as the solution to the ϵ-prox-mappings $\mathbf{P}_q^{\epsilon}(g)$, one based on weighted particles and the other based on weighted kernel density estimator. The first strategy is designed for the situation when the prior is a "good" guess of the true posterior, while the second strategy works for general situations. Interestingly, these two methods resemble particle reweighting and rejuvenation respectively in sequential Monte Carlo yet with notable differences. We will characterize the errors induced by these approximations in Section 4.4.

4.3.1 Posterior Approximation Using Weighted Particle

We first consider the situation when we are given a "good" prior, such that $p(\theta)$ has the same support as the true posterior $q^*(\theta)$, *i.e.*, $0 \leqslant q^*(\theta)/p(\theta) \leqslant C$. We will simply maintain a set of samples (or particles) from $p(\theta)$, and utlize them to estimate the intermediate prox-mappings. Let $\{\theta_i\}_{i=1}^m \sim p(\theta)$ be a set of fixed *i.i.d.* samples. We approximate $q_{t+1}(\theta)$ as a set of weighted particles

$$\begin{aligned} \tilde{q}_{t+1}(\theta) &= \textstyle\sum_{i=1}^m \alpha_i^{t+1}\, \delta(\theta_i), \\ \alpha_i^{t+1} &:= \tfrac{\alpha_i^t \exp(-\gamma_t g_t(\theta_i))}{\sum_{i=1}^m \alpha_i^t \exp(-\gamma_t g_t(\theta_i))}, \forall t \geqslant 1. \end{aligned} \tag{4.5}$$

The update is derived from the closed-form solution to the *exact* prox-mapping step (4.3). Since the normalization is a constant common to all components, one can simply update the set of working variable α_i as

$$\begin{aligned} \alpha_i &\leftarrow \alpha_i^{1-\gamma_t} p(x_t|\theta_i)^{N\gamma_t}, \forall i \\ \alpha_i &\leftarrow \frac{\alpha_i}{\sum_{i=1}^m \alpha_i}. \end{aligned} \tag{4.6}$$

We show that the one step approximation (4.5) incurs the error when estimating the integration of a function.

Theorem 16 *For any bounded and integrable function f,*

$$\mathbb{E}\left[\left|\int \tilde{q}_t(\theta)f(\theta)d\theta - \int q_t(\theta)f(\theta)d\theta\right|\right] \leqslant \frac{2C\|f\|_\infty}{\sqrt{m}}.$$

Remark: Please refer to the Appendix B.3 for details. When the model has several latent variables $\theta = (\xi, \zeta)$ and some parts of the variables have closed-form update in (4.3). *e.g.*, sparse GPs and LDA (refer to Appendix B.6), we could incorporate such structure information into algorithm by decomposing the posterior $q(\theta) = q(\xi)q(\zeta|\xi)$. When $p(\xi)$ satisfies the condition, we could sample $\{\xi_i\}_{i=1}^m \sim p(\xi)$ and approximate the posterior with summation of several functions, i.e., in the form of $q(\theta) \approx \sum \alpha_i q(\zeta|\xi_i)$.

4.3.2 Posterior Approximation Using Weighted Kernel Density Estimator

In general, sampling from prior $p(\theta)$ that are not so "good" will lead to particle depletion and inaccurate estimation of the posterior. To alleviate particle degeneracy, we propose to estimate the prox-mappings via weighted kernel density estimator (KDE). The weighted KDE prevents particles from dying out, in a similar fashion as kernel smoothing variant SMC [111] and one-pass SMC [92], but with guarantees.

More specifically, we approximate $q_{t+1}(\theta)$ via a weighted kernel density estimator

$$\begin{aligned} &\tilde{q}_{t+1}(\theta) = \sum\nolimits_{i=1}^m \alpha_i\, K_h(\theta - \theta_i), \\ &\alpha_i := \frac{\exp(-\gamma_t g_t(\theta_i))}{\sum_{i=1}^m \exp(-\gamma_t g_t(\theta_i))}, \quad \{\theta_i\}_{i=1}^m \overset{i.i.d.}{\sim} \tilde{q}_t(\theta), \end{aligned} \tag{4.7}$$

where $h > 0$ is the bandwidth parameter and $K_h(\theta) := \frac{1}{h^d}K(\theta/h)$ is a smoothing kernel. The update serves as an ϵ-prox-mapping (4.4) based on the closed-form solution to the *exact* prox-mapping step (4.3). Unlike the first strategy, the particle location in this case is sampled from the previous solution $\tilde{q}_t(\theta)$. The idea here is that $\tilde{q}_t^+(\theta) = \tilde{q}_t(\theta)\exp(-\gamma_t g_t(\theta))/Z$ can be viewed as an importance weighted version of $\tilde{q}_t(\theta)$ with weights equal to $\exp(-\gamma_t g_t(\theta))/Z$. If we want to approximate $\tilde{q}_t^+(\theta)$, we can sample m locations from $\tilde{q}_t(\theta)$ and associate each location the normalized weight α_i. To obtain a density for resampling in the next iteration, we place a kernel function $K_h(\theta)$ on each sampled location.

Since α_i is a ratio, we can avoid evaluating the normalization factor Z when computing α_i. In summary, we can simply update the set of working variable α_i as

$$\begin{aligned} \alpha_i &\leftarrow \tilde{q}_t(\theta_i)^{-\gamma_t} p(\theta_i)^{\gamma_t} p(x_t|\theta_i)^{N\gamma_t}, \forall i \\ \alpha_i &\leftarrow \frac{\alpha_i}{\sum_{i=1}^m \alpha_i}. \end{aligned} \tag{4.8}$$

Intuitively, the sampling procedure gradually adjusts the support of the intermediate distribution towards that of the true posterior, which is similar to "rejuvenation" step. The reweighting procedure gradually adjusts the shape of the intermediate distribution on the support. Same as the mechanism in [111, 92], the weighted KDE could avoid particle depletion.

We demonstrate that the estimator in (4.7) in one step possesses similar estimation properties as standard KDE for densities (for details, refer to the Appendix B.4).

Theorem 17 *Let q_t be a $(\beta; \mathcal{L})$-Hölder density function, and K be a β-valid density kernel, and the kernel bandwidth chosen as $h = O(m^{-\frac{1}{d+2\beta}})$. Then, under some mild conditions, $\mathbb{E}\left\|\tilde{q}_t(\theta) - q_t(\theta)\right\|_1 = O(m^{-\frac{\beta}{d+2\beta}})$.*

We specify the definition of $(\beta; \mathcal{L})$-Hölder family in the Appendix B.4. A kernel function $K(\cdot)$ is called β-valid, if $\int z^s K(z)dz = 0$ holds true for any $s = (s_1, \ldots, s_d) \in \mathbb{N}^d$ with $|s| \leqslant \lfloor\beta\rfloor$. Notice that all spherically symmetric and product kernels satisfy the condition. For instance, the Gaussian kernel $K(\theta) = (2\pi)^{-d/2} \exp(-\left\|\theta\right\|^2/2)$ satisfies the condition with $\beta = 1$, and it is used throughout our experiments. Theorem 17 implies that the weighted KDE achieves the *minmax* rate for density estimation in $(\beta; \mathcal{L})$-Hölder function class [112], where β stands for the smoothness parameter and $\mathcal{L}$ is the corresponding Lipschitz constant. With further assumption on the smoothness of the density, the weighted KDE can achieve even better rate. For instance, if β scales linearly with dimension, the error of weighted KDE can achieve a rate independent of the dimension.

Essentially, the weighted KDE step provides an ϵ-prox-mapping $\mathbf{P}_{\tilde{q}_t}^{\epsilon_t}(\gamma_t g_t)$ (4.4) in density space as we will discuss in Section 4.4.2. The inexactness is therefore determined by the number of samples m_t and kernel bandwidth h_t used in the weighted KDE.

4.3.3 Overall Algorithm

We present the overall algorithm, particle mirror descent (PMD), in Algorithm 3. The algorithm is based on stochastic mirror descent incorporated with two strategies from section 4.3.1 and 4.3.2 to compute prox-mapping. PMD takes as input N samples $X = \{x_n\}_{n=1}^N$, a prior $p(\theta)$ over the model parameter and the likelihood $p(x|\theta)$, and outputs the posterior density estimator $\tilde{q}_T(\theta)$ after T iterations. At each iteration, PMD takes the stochastic functional gradient information and computes an inexact prox-mapping $\tilde{q}_t(\theta)$ through either weighted particles or weighted kernel density estimator. Note that as discussed in Section 4.4, we can also take a batch of points at each iteration to compute the stochastic gradient in order to reduce variance.

Algorithm 3 Particle Mirror Descent Algorithm

Input: Data set $X = \{x_n\}_{n=1}^N$, prior $p(\theta)$
Output: posterior density estimator $\tilde{q}_T(\theta)$
Initialize $\tilde{q}_1(\theta) = p(\theta)$
for $t = 1, 2, \ldots, T-1$ **do**
  $x_t \overset{unif.}{\sim} X$
  if Good $p(\theta)$ is provided **then**
    $\{\theta_i\}_{i=1}^{m_t} \overset{i.i.d.}{\sim} p(\theta)$ when $t = 1$
    $\alpha_i \leftarrow \alpha_i^{1-\gamma_t} p(x_t|\theta_i)^{N\gamma_t}, \forall i$
    $\alpha_i \leftarrow \frac{\alpha_i}{\sum_{i=1}^{m_t} \alpha_i}, \forall i$
    $\tilde{q}_{t+1}(\theta) = \sum_{i=1}^{m_t} \alpha_i \, \delta(\theta_i)$
  else
    $\{\theta_i\}_{i=1}^{m_t} \overset{i.i.d.}{\sim} \tilde{q}_t(\theta)$
    $\alpha_i \leftarrow \tilde{q}_t(\theta_i)^{-\gamma_t} p(\theta_i)^{\gamma_t} p(x_t|\theta_i)^{N\gamma_t}, \forall i$
    $\alpha_i \leftarrow \frac{\alpha_i}{\sum_{i=1}^{m_t} \alpha_i}, \forall i$
    $\tilde{q}_{t+1}(\theta) = \sum_{i=1}^{m_t} \alpha_i K_{h_t}(\theta - \theta_i)$
  end if
end for

In Section 4.4, we will show that, with proper setting of stepsize γ, Algorithm 3 converges in rate $O(1/\sqrt{m})$ using m particles, in terms of either integral approximation or KL-divergence, to the true posterior.

In practice, we could combine the proposed two algorithms to reduce the computation cost. In the beginning stage, we adopt the second strategy. The computation cost is affordable for small number of particles. After we achieve a reasonably good estimator of

the posterior, we could switch to the first strategy using large size particles to get better rate.

4.4 Theoretical Analysis

In this section, we show that PMD algorithm (**i**) given good prior $p(\theta)$, achieves a sublinear rate of convergence in terms of integral approximation; and (**ii**) in general cases, achieves a dimension dependent, sublinear rate of convergence in terms of KL-divergence with proper choices of stepsizes.

4.4.1 Weak Convergence of PMD

The weighted particles approximation, $\tilde{q}_t(\theta) = \sum_{i=1}^m \alpha_i \, \delta(\theta_i)$, returned by Algorithm 3 can be used directly for Bayesian inference. That is, given a function f, $\int q^*(\theta) f(\theta) d\theta$ can be approximated as $\sum_{i=1}^m \alpha_i f(\theta_i)$. We will analyze its ability in approximating integral, which is commonly used in sequential Monte Carlo for dynamic models [113] and stochastic Langevin dynamics [99]. For simplicity, we may write $\sum_{i=1}^m \alpha_i f(\theta_i)$ as $\int \tilde{q}_t(\theta) f(\theta) d\theta$, despite of the fact that $\tilde{q}_t(\theta)$ is not exactly a density here. We show a sublinear rate of convergence in terms of integral approximation.

Theorem 18 (Integral approximation) *Assume $p(\theta)$ has the same support as the true posterior $q^*(\theta)$,* i.e., *$0 \leqslant q^*(\theta)/p(\theta) \leqslant C$. Assume further model $\|p(x|\theta)^N\|_\infty \leqslant \rho, \forall x$. Then $\forall f(\theta)$ bounded and integrable, the T-step PMD algorithm with stepsize $\gamma_t = \frac{\eta}{t}$ returns m weighted particles such that*

$$\mathbb{E}\left[\left|\int \tilde{q}_T(\theta) f(\theta) d\theta - \int q^*(\theta) f(\theta) d\theta\right|\right] \leqslant \frac{2\sqrt{\max\{C, \rho e^M)\}}\|f\|_\infty}{\sqrt{m}} + \max\left\{\sqrt{KL(q^*||p)}, \frac{\eta M}{\sqrt{2\eta - 1}}\right\} \frac{\|f\|_\infty}{\sqrt{T}}$$

where $M := \max_{t=1,\ldots,T} \|g_t\|_\infty$.

Remark: The condition for the models, $\|p(x|\theta)^N\|_\infty \leqslant \rho, \forall x$, is mild, and there are plenty of models satisfying the requirement. For examples, in binary/multi-class logistic regression,

probit regression, as well as latent Dirichlet analysis, $\rho \leqslant 1$. Please refer to details in Appendix B.3. It should be pointed out that the constant C can indirectly depends on the dimension of θ. The concrete dependence between C and d is related to the property of particular models and out of the scope of this thesis. The proof combines the results of the weighted particles for integration, and convergence analysis of mirror descent. One can see that the error consists of two terms, one from integration approximation and the other from optimization error. To achieve the best rate of convergence, we need to balance the two terms. That is, when the number particles, m, scales linearly with the number of iterations, we obtain an overall convergence rate of $O(\frac{1}{\sqrt{T}})$. In other words, if the number of particles is fixed to m, we could achieve the convergence rate $O(\frac{1}{\sqrt{m}})$ with $T = O(m)$ iterations.

4.4.2 Strong Convergence of PMD

In general, when the weighted kernel density approximation scheme is used, we show that PMD enjoys a much stronger convergence, *i.e.*, the KL-divergence between the generated density and the true posterior converges sublinearly. Throughout this section, we merely assume that

Assumption 5 *The prior and likelihood are bounded and belong to* $(\beta; \mathcal{L})$*-Hölder class.*

Assumption 6 *Kernel* $K(\cdot)$ *is a* β*-valid density kernel with a compact support and there exists* $C_k, \mu, \nu > 0$ *such that* $\|K(\cdot)\|_\infty \leqslant C_k$, $\int K(z)^2\, dz \leqslant \mu^2$, $\int \|z\|^\beta |K(z)| dz \leqslant \nu$.

Assumption 7 *There exists a bounded support* Ω *such that the* log *function is* Δ*-Lipschitz continuous.*

Note that the above assumptions are more of a brief characteristics of the commonly used kernels and inferences problems in practice rather than an exception. The first condition clearly holds true when the logarithmic of the prior and likelihood belongs to $\mathcal{C}_\infty$ with bounded derivatives of all orders, as assumed in several literature [98, 99]. The third condition is for characterizing the estimator over its support. These assumptions automatically validate all the conditions required to apply Theorem 17 and the corresponding

high probability bounds (stated in Corollary 50 in appendix). Let the kernel bandwidth $h_t = m_t^{-1/(d+2\beta)}$, we immediately have that with high probability,

$$\|\tilde{q}_{t+1} - \mathbf{P}_{\tilde{q}_t}(\gamma_t g_t)\|_1 \leqslant O(m_t^{-\beta/(d+2\beta)}).$$

Directly applying Theorem 15, and solving the recursion following [70], we establish the convergence results in terms of KL-divergence.

Theorem 19 (KL-divergence) *Based on the above assumptions, when setting* $\gamma_t = O\left(\frac{1}{t}\right)$,

$$\mathbb{E}[KL(q^*||\tilde{q}_T)] \quad \leqslant \quad \frac{2\max\left\{D_1, M^2\right\}}{T} + \mathcal{C}_1 \frac{\sum_{t=1}^{T} t^2 m_t^{-\frac{\beta}{d+2\beta}}}{T^2} + \mathcal{C}_2 m_T^{-\frac{\beta}{d+2\beta}}$$

where $M := \max_{t=1,\ldots,T} \|g_t\|_\infty$, $D_1 = KL(q^*||\tilde{q}_1)$, $\mathcal{C}_1 := O(1)\Delta C_k\left(\nu\mathcal{L} + \mu\right)$, *and* $\mathcal{C}_2 := O(1)(\mu + \nu\mathcal{L})$ *with* $O(1)$ *being a constant.*

Remark: Unlike Theorem 18, the convergence results are established in terms of the KL-divergence, which is a stronger criterion and can be used to derive the convergence under other divergences [114]. To our best knowledge, these results are the first of its kind for estimating posterior densities in literature. One can immediately see that the final accuracy is essentially determined by two sources of errors, one from noise in applying stochastic gradient, the other from applying weighted kernel density estimator. For the last iterate, an overall $O(\frac{1}{T})$ convergence rate can be achieved when $m_t = O(t^{2+d/\beta})$. There is an explicit trade-off between the overall rate and the total number of particles: the more particles we use at each iteration, the faster algorithm converges. One should also note that in our analysis, we explicitly characterize the effect of the smoothness of model controlled by β, which is assumed to be infinite in existing analysis of SGLD. When the smoothness parameter $\beta >> d$, the number of particles no longer depends on the dimension. That means, with memory budget $O(dm)$, *i.e.*, the number of particles is set to be $O(m)$, we could achieve a $O(1/\sqrt{m})$ rate.

Table 4.1: Summary of the related approximate inference methods

Methods	Provable	Convergence Criterion	Convergence Rate	Cost		Black Box
				Computation per Iteration	Memory	
SVI	No	–	–	$\Omega(d)$	$O(d)$	No
NPV	No	–	–	$\Omega(dm^2N + d^2N)$	$O(dm)$	No
Static SMC	No	–	–	$\Omega(dm)$	$O(dm)$	Yes
SGLD	Yes	$\lvert\langle q - q^*, f\rangle\rvert$	$O(m^{-\frac{1}{3}})$	$\Omega(d)$	$O(dm)$	Yes
PMD	Yes	$\lvert\langle q - q^*, f\rangle\rvert$	$O(m^{-\frac{1}{2}})$	$\Omega(dm)$	$O(dm)$	Yes
		$KL(q^*\lvert\lvert q)$	$O(m^{-\frac{1}{2}})$	$\Omega(dm^2)$	$O(dm)$	

4.5 Connections to Other Approaches

PMD connects stochastic optimization, Monte Carlo approximation and functional analysis to Bayesian inference. Therefore, it is closely related to two different paradigms of inference algorithms derived based on either optimization or Monte Carlo approximation.

Relation to SVI From the optimization point of view, the proposed algorithm shares some similarities to stochastic variational inference (SVI) [106]–both algorithms utilize stochastic gradients to update the solution. However, SVI optimizes a *surrogate of the objective*, the evidence lower bound (ELBO), with respect to a *restricted parametric* distribution[1]; while the PMD directly optimizes the objective over *all valid densities* in a nonparametric form. Our flexibility in density space eliminates the bias and leads to favorable convergence results.

Relation to SMC From the sampling point of view, PMD and the particle filtering/sequential Monte Carlo (SMC) [111] both rely on importance sampling. In the framework of SMC sampler [115], the static SMC variants proposed in [91, 92] bares some resemblances to the proposed PMD. However, their updates come from completely different origins: the static SMC update is based on Monte Carlo approximation of Bayes' rule, while the PMD update based on inexact prox-mappings. On the algorithmic side, (**i**) the static SMC re-weights the particles with likelihood while the PMD re-weights based on functional gradient, which can be fractional power of the likelihood; and (**ii**) the static SMC only utilizes each datum once

[1]Even in [108], "nonparametric variational inference" (NPV) uses the mixture of Gaussians as variational family which is still parametric.

while the PMD allows multiple pass of the datasets. Most importantly, on the theoretical side, PMD is guaranteed with convergence in terms of both KL-divergence and integral approximation for *static model*, while SMC is only rigoriously justified for *dynamic models*. It is unclear whether the convergence still holds for these extensions.

Summary of the comparison We summarize the comparison between PMD and static SMC, SGLD, SVI and NPV in Table 4.1. For the connections to other inference algorithms, including Annealed IS [116], general SMC sampler [115], stochastic gradient dynamics family [93, 95, 97, 96], nonparametric variational inference [117, 118, 119, 108, 120], as well as one following work of our PMD algorithm [121], please refer to Appendix B.7. Given dataset $\{x_i\}_{i=1}^N$, the model $p(x|\theta)$, $\theta \in \mathbb{R}^d$ and prior $p(\theta)$, whose value and gradient could be computed, we set PMD, static SMC, SGLD and NPV to keep m samples/components, so that they have the same memory cost and comparable convergence rate in terms of m. Therefore, SGLD runs $O(m)$ iterations. Meanwhile, by balancing the optimization error and approximation in PMD, we have PMD running $O(m)$ for integal approximation and $O(\sqrt{m})$ for KL-divergence. For static SMC, the number of iteration is $O(N)$. From Table 4.1, we can see that there exists a delicate trade-off between computation, memory cost and convergence rate for the approximate inference methods.

1. The static SMC uses simple normal distribution [91] or kernel density estimation [92] for rejuvenation. However, such a moving kernel is purely heuristic and it is unclear whether the convergence rate of SMC for dynamic system [113, 122] still holds for static models. To ensure the convergence of static SMC, MCMC is needed in the rejuvenation step. The MCMC step requires to browse all the previously visited data, leading to extra computation cost $\Omega(dmt)$ and memory cost $O(dt)$, and hence violating the memory budget requirement. We emphasize that even using MCMC in static SMC for rejuvenation, the conditions required for static SMC is more restricted. We discuss the conditions for convergence of SMC and PMD using particles approximation in Appendix B.3.

2. Comparing with SGLD, the cost of PMD at each iteration is higher. However, PMD

converges in rate of $O(m^{-\frac{1}{2}})$, faster than SGLD, $O(m^{-\frac{1}{3}})$, in terms of integral approximation and KL-divergence which is more stringent if *all the orders* of derivatives of stochastic gradient is bounded. Moreover, even for the integral approximation, SGLD converges only when f having weak Taylor series expansion, while for PMD, f is only required to be bounded. The SGLD also requires the stochastic gradient satisfying several extra conditions to form a Lyapunov system, while such conditions are not needed in PMD.

4.6 Experiments

We conduct experiments on mixture models, logistic regression, sparse Gaussian processes and latent Dirichlet allocation to demonstrate the advantages of PMD in capturing multiple modes, dealing with non-conjugate and hierarchical models, respectively.

Competing algorithms For the mixture model and logistic regression, we compare our algorithm with five general approximate Bayesian inference methods, including three sampling algorithms, *i.e.*, one-pass sequential Monte Carlo (one-pass SMC) [92] which is an improved version of the SMC for Bayesian inference [91], stochastic gradient Langevin dynamics (SGD Langevin) [93] and Gibbs sampling, and two variational inference methods, *i.e.*, stochastic variational inference (SVI) [106] and stochastic variant of nonparametric variational inference (SGD NPV) [108]. For sparse GP and LDA, we compare with the existing large-scale inference algorithms designed specifically for the models.

Evaluation criterion For the synthetic data generated by mixture model, we could calculate the true posterior, Therefore, we evaluate the performance directly through total variation and KL-divergence (cross entropy). For the experiments on logistic regression, sparse GP and LDA on real-world datasets, we use indirect criteria which are widely used [96, 97, 123, 124, 106] because of the intractability of the posterior. We keep the same memory budget for Monte Carlo based algorithms if their computational cost is acceptable. To demonstrate the efficiency of each algorithm in utilizing data, we use the number of data visited cumulatively as x-axis.

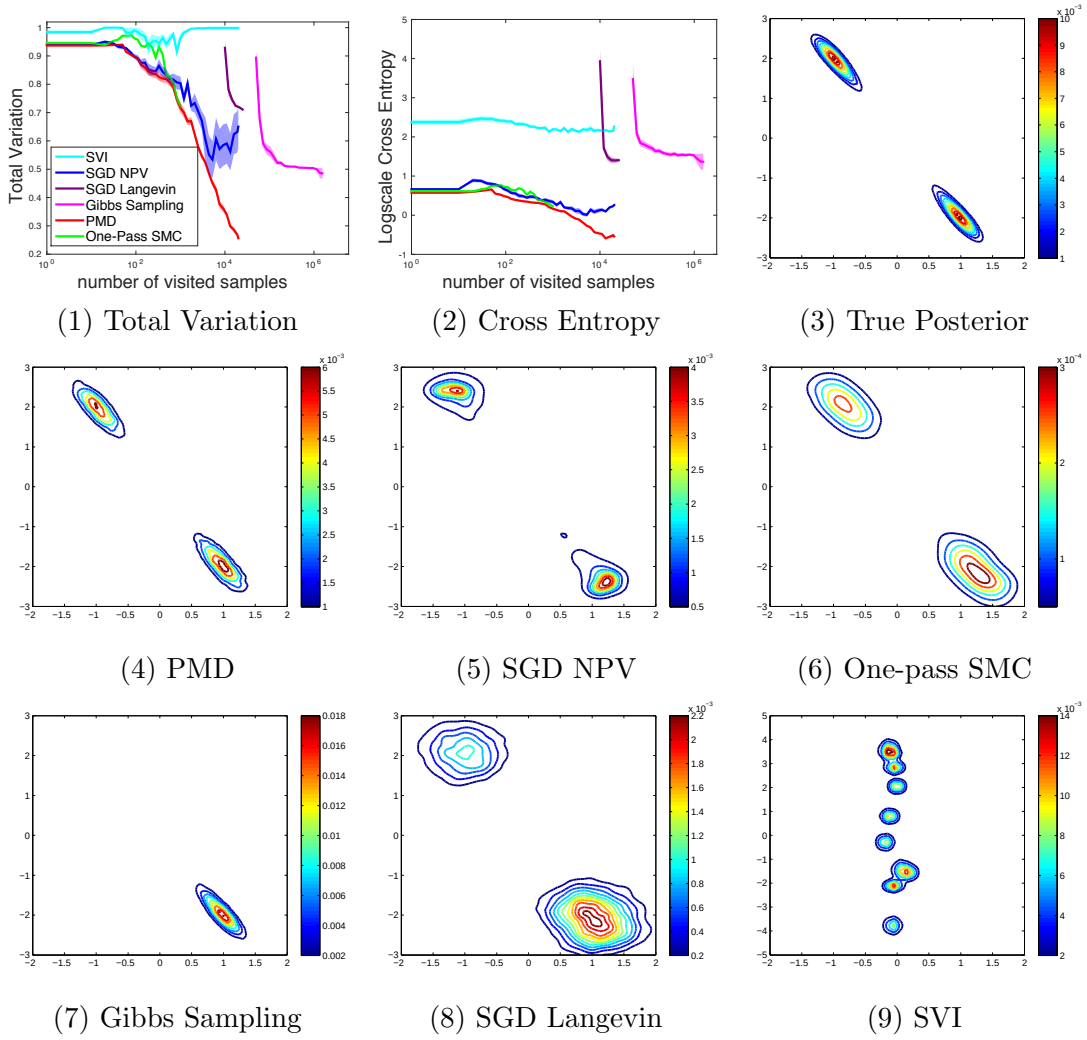
For additional results and algorithm derivations for sparse GP and LDA, please refer to the Appendix B.8.

4.6.1 Mixture Models

We conduct comparison on a simple yet interesting mixture model [93], the observations $x_i \sim p\mathcal{N}(\theta_1, \sigma_x^2) + (1-p)\mathcal{N}(\theta_1 + \theta_2, \sigma_x^2)$ and $\theta_1 \sim \mathcal{N}(0, \sigma_1^2)$, $\theta_2 \sim \mathcal{N}(0, \sigma_2^2)$, where $(\sigma_1, \sigma_2) = (1, 1)$, $\sigma_x = 2.5$ and $p = 0.5$. The means of two Gaussians are tied together making θ_1 and θ_2 correlated in the posterior. We generate 1000 data from the model with $(\theta_1, \theta_2) = (1, -2)$. This is one mode of the posterior, there is another equivalent mode at $(\theta_1, \theta_2) = (-1, 2)$.

We initialize all algorithms with prior on (θ_1, θ_2). We use the normalized Gaussian kernel in this experiment. For one-pass SMC, we use the suggested kernel bandwidth in [92]. For our method, since we increase the samples, the kernel bandwidth is shrunk in rate of $O(m^{-\frac{1}{2}})$ as the theorem suggested. The batch size for stochastic algorithms and one-pass SMC is set to be 10. The total number of particles for the Monte Carlo based competitors, *i.e.*, SMC, SGD Langevin, Gibbs sampling, and our method is 1500 in total. We also keep 1500 Gaussian components in SGD NPV. The burn-in period for Gibbs sampling and stochastic Langevin dynamics are 50 and 1000 respectively. We repeat the experiments 10 times and report the average results. We keep the same memory for all except SVI.

To compare these different kinds of algorithms in a fair way, we evaluate their performances using total variation and cross entropy of the solution against the true potential functions versus the number of observations visited. In order to evaluate the total variation and the cross entropy between the true posterior and the estimated one, we use both kernel density estimation and Gaussian estimation to approximate the posterior density and report the better one for Gibbs sampling and stochastic Langevin dynamics. The kernel bandwidth is set to be 0.1 times the median of pairwise distances between data points (median trick). PMD achieves the best performance in terms of total variation and cross entropy as shown in Figure 4.1 (1)(2). The one-pass SMC performs similar to our algorithm at beginning. However, it cannot utilize the dataset effectively, therefore, it stopped with high error. It should be noticed that the one-pass SMC starts with more particles while our algorithm only requires the same number of particles at final stage. The reason



(1) Total Variation (2) Cross Entropy (3) True Posterior

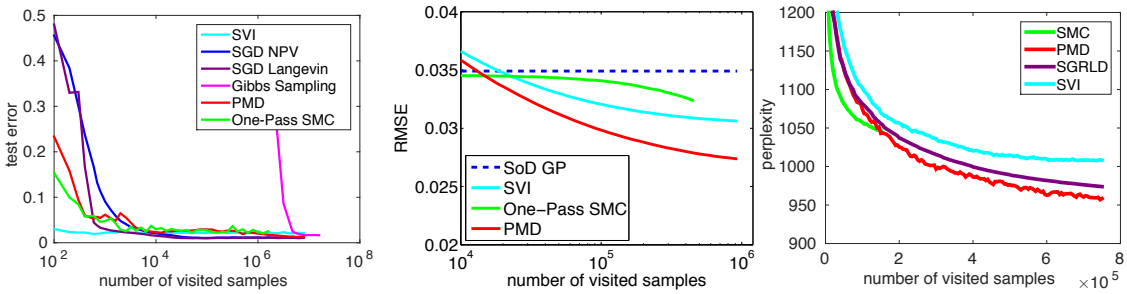
(4) PMD (5) SGD NPV (6) One-pass SMC

(7) Gibbs Sampling (8) SGD Langevin (9) SVI

Figure 4.1: Experimental results for mixture model on synthetic dataset. Figure (1) (2) shows the comparison between PMD and the competitors using total variation and cross entropy, respectively. Figures (3)–(9) are the visualization of posteriors of mixture model on synthetic dataset obtained by several inference methods.

that Gibbs sampling and the stochastic gradient Langevin dynamics perform worse is that they stuck in one mode. It is reasonable that Gibbs sampling fits the single mode better than stochastic gradient Langevin dynamics since it generates one new sample by scanning the whole dataset. For the stochastic nonparametric variational inference, it could locate both modes, however, it optimizes a non-convex objective which makes its variance much larger than our algorithm. The stochastic variational inference fails because of the highly dependent variables and multimodality in posterior.

The true posterior is illustrated in Figure 4.1 (3) and the results generated by PMD



(1) Logistic regression on MNIST (2) Sparse GP on music data (3) LDA on wikipedia data

Figure 4.2: Experimental results on several different models for real-world datasets.

and the competitors are illustrated in Figure 4.1 (4)–(9). From these figures, we could have a direct understand about the behaviors for each competitors. PMD fits both modes well and recovers nicely the posterior while other algorithms either miss a mode or fail to fit the multimodal density. The Gibbs sampling and stochastic gradient Langevin dynamics sampling stuck in one local mode in each run. Gibbs sampler could fit one of the contour quite well, better than the stochastic Langevin dynamics. It should be noticed that this is the average solution, the two contours in the result of stochastic gradient Langevin dynamics did not mean it finds both modes simultaneously. The one-pass sequential Monte Carlo and stochastic nonparametric variational inference are able to location multiple modes. However, their shapes are not as good as ours. Because of the multiple modes and the highly dependent variables in posterior, the stochastic variational inference fails to converge to the correct modes.

This experiment clearly indicates our algorithm is able to take advantages of nonparametric model to capture multiple modes.

4.6.2 Bayesian Logistic Regression

We test our algorithm on logistic regression with non-conjugate prior for handwritten digits classification on the MNIST8M 8 vs. 6 dataset. The likelihood function in logistic regression is $p(y|x,w) = \frac{1}{1+\exp(-yw^\top x)}$ with w as the latent variables. We use Gaussian prior for w with identity covariance matrix.

The dataset contains about 1.6M training samples and 1932 testing samples. We first reduce the dimension to 50 by PCA. The batch size is set to be 100 and the step size is

set to be $\frac{1}{100+\sqrt{t}}$. We initialize all algorithms with same prior and terminate the stochastic algorithms after 5 passes through the dataset. The burn-in period for stochastic Langevin dynamic is set to be 1000. We rerun the experiments 10 times. We keep 1000 samples for Monte Carlo based algorithms, except Gibbs sampling whose computation cost is unaffordable. We repeat the experiments 10 times and the results are reported in Figure 4.2(1).

Obviously, Gibbs sampling [125], which needs to scan the whole dataset, is not suitable for large-scale problem. In this experiment, SVI performs best at first, which is expectable because learning in the Gaussian family is simpler comparing to nonparametric density family. Our algorithm achieves comparable performance in nonparametric form after fed with enough data, 98.8%, to SVI which relies on carefully designed lower bound of the log-likelihood [126]. SGD NPV is flexible with mixture models family. Although the stochastic variant of nonparametric variational inference performs comparable to our algorithm with fewer components, its speed is bottleneck when applied to large-scale problems. The gain from using stochastic gradient is dragged down by using L-BFGS to optimize the second-order approximation of the evidence lower bound.

4.6.3 Sparse Gaussian Processes

We first tested the PMD on sparse GP on 1D synthetic data which is illustrated in Appendix B.8. We use sparse GPs models to predict the year of songs [127]. In this task, we compare to the SVI for sparse GPs [106, 123]and one-pass SMC. We also included subset of data approximation (SoD) [128] as baseline. We randomly selected $463,715$ songs to train the model and test on $5,163$ songs, each represented by 90-dimension features. As in [127], the year values are linearly mapped into $[0, 1]$. The data is standardized before regression. Gaussian RBF kernel is used in the model. Since we are comparing the inference algorithms, for fairness, we fixed the model hyperparameters for all the inference algorithms, *i.e.*, the kernel bandwidth is set to be the median of pairwise distances between data points and the observations precision $\beta^{-1} = 0.01$. We set the number of inducing inputs to be 2^{10} and batch size to be 512. The stepsize for both PMD and SVI are in the form of $\frac{\eta}{n_0+\sqrt{t}}$. To demonstrate the advantages of PMD comparing to SMC, we initialize PMD with prior while SMC with the SoD solution. We terminate the stochastic algorithms after 2 passes

of dataset. We use 16 particles in both SMC and PMD. We run experiments 10 times and results are reported in Figure. 4.2(2). Our algorithm achieves the best RMSE 0.027, significantly better than one-pass SMC and SVI.

4.6.4 Latent Dirichlet Allocation

We compare to SVI [106], stochastic gradient Riemannian Langevin dynamic (SGRLD) [124], and SMC specially designed for LDA [129] on Wikipedia dataset [124]. The dataset contains 0.15M documents, about 2M words and 8000 vocabulary. Since we evaluate their performances in terms of perplexity, which is integral over posterior, we do not need to recover the posterior, and therefore, we follow the same setting in [130, 131], where one particle is used in SMC and PMD to save the cost. We set topic number to 100 and fix other hyperparameters $\alpha = 0.1$ and $\beta = 0.01$ to be fair to all algorithms. The batchsize is set to be 100. We use stepsize $\frac{\eta}{n_0+t^\kappa}$ for PMD, stochastic variational inference and stochastic Riemannian Langevin dynamic. For each algorithm a grid-search was run on step-size parameters and the best performance is reported. We stop the stochastic algorithms after they pass through the whole dataset 5 times. The results are reported in Figure 4.2(3). The top words from several topics found by our algorithm are illustrated in Appendix B.8. Our algorithm achieves the best perplexity, significantly better than SGRLD and SVI. In this experiment, SMC performs well at the beginning since it treats each documents equally and updates with full likelihood. However, SMC only uses each datum once, while the stochastic algorithms, *e.g.*, SGRLD, SVI and our PMD, could further refine the solution by running the dataset multiple times.

4.7 Summary

The integral operator view of Bayes' rule recasts Bayesian inference as the special case of the unified framework, *i.e.*, learning over distributions problems. From such a new perspective, we can avoid the general intractable integral by an optimization, so that we leverage the advances from optimization algorithms and sampling technique towards achieving better trade-off between *efficiency*, *flexibility* and *provability* in approximate Bayesian inference.

The proposed particle mirror descent algorithm successfully combines stochastic mirror descent and nonparametric density approximation. Theoretically, the algorithm enjoys a rate $O(1/\sqrt{m})$ in terms of both integral approximation and KL-divergence, with $O(m)$ particles. Practically, the algorithm achieves competitive performance to existing state-of-the-art inference algorithms in mixture models, logistic regression, sparse Gaussian processes and latent Dirichlet analysis on several large-scale datasets.

PART III: PART III: LEARNING OVER DYNAMICS

In this part, we will discuss the problem that each sample x itself is associated with a conditional distribution $p(z|x)$ represented by samples $\{z_i\}_{i=1}^M$, and the goal is to learn a function f that links these conditional distributions to target values y. In fact, this problem is also a special case of the proposed framework (2.3). Specifically, we instantiate the Eq (2.3) with conditional integral operator, define as

$$(\mathcal{T} f)(x) := \langle p(z|x), f(z,x) \rangle : \mathcal{F}(\mathcal{X} \times \mathcal{Z}) \to \mathcal{F}$$

and $G(\cdot)$ as some optional norms of f, then, we obtain the optimization

$$\min_{f \in \mathcal{F}} \ \mathbb{E}_{x,y} \left[J\left(\mathbb{E}_{z|x} \left[f(\cdot, z) \right], (x, y) \right) \right] + \nu G(f). \tag{4.9}$$

In dynamics learning settings, *e.g.*, reinforcement learning and time series, the inputs, *i.e.*, $p(z|x)$, fully characterize the transition of the dynamics, therefore, such a problem can be understood as *learning over dynamics*.

This learning problem appears in many tasks, *e.g.*, invaraince learning, policy evaluation in reinforcement learning, events prediction in stochastic processes, and so on. The integral operator view provides us a unified view connecting the tasks which are investigated separately before, and thus, enables us to develop a general and efficient algorithm, *embedding-SGD*, which can be applied to these tasks in Chapter 5.

We further discuss the policy optimization problem in Chapter 6, which is a difficult generalization of the optimization with the conditional integral operator (4.9), by applying the dual embedding technique to the mean squared consistency Bellman error. We obtain a novel *off-policy* reinforcement learning which demonstrates state-of-the-art performance empirically and can be proved convergent with nonlinear function approximation.

CHAPTER 5
DUAL EMBEDDING FOR CONDITIONAL INTEGRAL OPERATOR

The invarance learning [27], policy evaluation in reinforcement learning [28, 29], as well as several other tasks [132, 133] are imporant machine learning problems, which have been investigated separately for decades. With the proposed framework, we can unify these problems as a special case by plugging the coniditional integral operator into the framework (2.3),

$$\min_{f \in \mathcal{F}} \; L(f) = \mathbb{E}_{x,y} \left[\ell \left(\mathbb{E}_{z|x} \left[f(z,x) \right], y \right) \right] + \nu G(f) \tag{5.1}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a convex loss function. Such an abstract view unifies these tasks, and thus, inspires the general *dual embedding* technique which can be applied to multiple applications. The learning problem become very challenging when we only have limited samples or in the extreme case only one sample from each conditional distribution, *e.g.*, reinforcement learning. The proposed dual embedding technique successfully address the sampling challenge by employing a new min-max reformulation of Eq (5.1). It leads to the *embedding-SGD* algorithm which only needs to deal with the joint distribution $p(z, x)$. We provide the theoretical analysis for the proposed general algorithm, and test the algorithm on several applications, including invariance learning and policy evaluation, demonstrating the advantages of the proposed algorithm.

5.1 Introduction

We address the problem of *learning with dynamics* (5.1), in which the goal is to learn a function that links conditional distributions, which fully characterize the transition of a dynamics, to target variables. Specifically, we are provided input samples $\{x_i\}_{i=1}^N \in \mathcal{X}^N$ and their corresponding responses $\{y_i\}_{i=1}^N \in \mathcal{Y}^N$. For each $x \in \mathcal{X}$, there is an associated conditional distribution $p(z|x) : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$. However, we cannot access the entire conditional distributions $\{p(z|x_i)\}_{i=1}^N$ directly; rather, we only observe a limited number of samples or in the extreme case only *one sample* from each conditional distribution $p(z|x)$.

The function space $\mathcal{F}$ can be very general, but we focus on the case when $\mathcal{F}$ is constructed by a RKHS (partially) in main text, namely, $\mathcal{F} = \{f : \mathcal{Z} \times \mathcal{X} \to \mathbb{R} \,|\, f(z,x) = \langle f, \psi(z,x)\rangle\}$ or $\mathcal{F}_g = \{g + f : \mathcal{Z} \times \mathcal{X} \to \mathbb{R} \,|\, f(z,x) = \langle f, \psi(z,x)\rangle\}$ where $\psi(z,x)$ is a suitably chosen (nonlinear) feature map. It can be extended to arbitrary function approximators, *e.g.*, random features and neural networks, in Appendix C.5. The problem of optimization with conditional integral operator in (5.1) appears in many different tasks. For example:

- **Learning with invariance** Incorporating priors on invariance into the learning procedure is crucial for computer vision [134], speech recognition [135] and many other applications. The goal of invariance learning is to estimate a function which minimizes the expected risk while at the same time preserving consistency over a group of operations $g = \{g_j\}_{j=1}^{\infty}$. [27] shows that this can be accomplished by solving the following optimization problem

$$\min_{f \in \tilde{\mathcal{H}}} \mathbb{E}_{x,y}[\ell(\mathbb{E}_{z|x\sim\mu(g(x))}[\langle f, \psi(z)\rangle_{\tilde{\mathcal{H}}}], y)] + (\nu/2)\|f\|^2_{\tilde{\mathcal{H}}} \tag{5.2}$$

  where $\tilde{\mathcal{H}}$ is the RKHS corresponding to kernel $\tilde{k}$ with implicit feature map $\psi(\cdot)$, $\nu > 0$ is the regularization parameter. Obviously, the above optimization (5.2) is a special case of (5.1). In this case, z stands for possible variations of data x through conditional probability given by some normalized Haar measure $\mu(g(x))$. Due to computation and memory constraints, one can only afford to generate a few virtual samples from each data point x.

- **Policy evaluation in reinforcement learning** Policy evaluation is a fundamental task in reinforcement learning. Given a policy $\pi(a|s)$ which is a distribution over action space condition on current state s, the goal is to estimate the value function $V^\pi(\cdot)$ over the state space. $V^\pi(s)$ is the fixed point of the Bellman equation

$$V^\pi(s) = \mathbb{E}_{s',a|s}[R(s,a) + \gamma V^\pi(s')],$$

  where $R(s,a) : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is a reward function with $\mathcal{S}$ and $\mathcal{A}$ denoting the state and action space, respectively, $\gamma \in (0,1)$ is the discount factor, and $\mathbb{E}_{s',a|s'}$ denotes the expectation

w.r.t. $p\left(s'|s,a\right)\pi\left(a|s\right)$. Therefore, the value function can be estimated from data by minimizing the mean-square Bellman error [28, 29]:

$$\min_{V^\pi\in\mathcal{H}}\ \mathbb{E}_s\left[\left(\mathbb{E}_{s',a|s}\left[R(s,a)+\gamma V^\pi(s')-V^\pi(s)\right]\right)^2\right]+\nu\left\|V^\pi\right\|_{\mathcal{H}}^2. \tag{5.3}$$

Restrict the policy to lie in some RKHS $\mathcal{H}$, this optimization is clearly a special case of (5.1) by viewing $(s,0,(s',a))$ as (x,y,z) and $f\left(s,a,s'\right)=R(s,a)+\gamma V^\pi(s')-V^\pi(s)$ in (5.1). Here, given state s and the the action $a\sim\pi(a|s)$, the successor state s' comes from the transition probability $p(s'|a,s)$. Due to the online nature of MDPs, we usually observe only one successor state s' for each action a given s, *i.e.*, only one sample from the conditional distribution given s.

- **Optimal control in linearly-solvable MDP** The optimal control in a certain class of MDP, *i.e.*, linearly-solvable MDP, can be achieved by solving the linear Bellman equation [132, 133]

$$z(s)=\exp\left(-R(s)\right)\mathbb{E}_{s'|s\sim p(s'|s)}\left[z(s')\right], \tag{5.4}$$

where $R(s)$ denotes the immediate cost and $p(s'|s)$ denotes the passive dynamics without control. With $z(s)$, the trajectory of the optimal control π^* can be calculated by $p^{\pi^*}(s'|s)=\frac{p(s'|s)z(s)}{\mathbb{E}_{s'|s\sim p(s'|s)}[z(s')]}$. Therefore, $z(\cdot)$ can be estimated from data by optimizing

$$\min_{z\in\mathcal{H}}\ \mathbb{E}_s\left[\left(\mathbb{E}_{s'|s}\left[z(s)-\exp\left(-R(s)\right)z(s')\right]\right)^2\right]+\nu\left\|z\right\|_{\mathcal{H}}^2. \tag{5.5}$$

Restricting function $z(\cdot)$ to lie in some RKHS $\mathcal{H}$, this optimization is a special case of (5.1) by viewing $(s,0,s')$ as (x,y,z) in (5.1). Here given a state s, the successor state s' comes from the passive dynamics. Similar as policy evaluation, we usually observe only one successor state s' given s, *i.e.*, only one sample from the conditional distribution given s.

- **Hitting time and stationary distribution in stochastic process** Estimating the hitting time and stationary distribution of stochastic process are both important problems in social network application and MCMC sampling technique. Denote the transition probability as $p(s'|s)$. The hitting time of s_j starting from s_i is defined as $H(s_i,s_j)=$

$\inf\{n \geqslant 0; S_n = s_j, S_0 = s_i\}$. Hence, the expected hitting time $h(\cdot,\cdot)$ satisfies

$$h(s_i, s_j) = \begin{cases} 1 + \mathbb{E}_{s_k \sim p(s|s_i), s_k \neq s_j}\left[h(s_k, s_j)\right] & \text{if } i \neq j \\ 1 + \mathbb{E}_{s_k \sim p(s|s_i)}\left[h(s_k, s_i)\right] & \text{if } i = j \end{cases}. \tag{5.6}$$

Based on the property of stochastic process, we can obtain the stationary distribution with $\pi(s_i) = \frac{1}{h(s_i,s_i)}$. The hitting time $h(\cdot,\cdot)$ can be learned by minimizing:

$$\min_{h \in \mathcal{H}} \mathbb{E}_{s,t}\left[\left(1 - \mathbb{E}_{s' \sim \tilde{p}(s'|s,t)}\left[h(s,t) - h(s',t)\right]\right)^2\right] + \nu \left\|h\right\|_{\mathcal{H}}^2, \tag{5.7}$$

where $\tilde{p}(s'|s,t) = p(s'|s)$ if $s = t$, otherwise $\tilde{p}(s'|s,t) \propto \begin{cases} p(s'|s) & \text{if } s' \neq t \\ 0 & \text{if } s' = t \end{cases}$. Similarly, when restricting the expected hitting time to lie in some RKHS $\mathcal{H}$, this optimization is a special case of (5.1) by viewing $((s,t),1,s')$ as (x,y,z) in (5.1). Due to the stochasticity of the process, we only obtain one successor state s' from current state (s,t), *i.e.*, only one sample from the conditional distribution given (s,t).

Challenges Despite the prevalence of learning problems in the form of (5.1), solving such problems remain very challenging for two reasons: (i) we often have limited samples or in the extreme case only one sample from each conditional distribution $p(z|x)$, making it difficult to accurately estimate the conditional expectation. (ii) the conditional expectation is nested inside the loss function, making the problem quite different from the traditional stochastic optimization setting. This type of problem is called *compositional stochastic programming*, and very few results have been established in this domain.

5.1.1 Related Work

A simple option to address (5.1) is using sample average approximation (SAA), and thus, instead solve

$$\min_{f \in \mathcal{F}} \frac{1}{N} \sum_{i=1}^{N} \left[\ell\left(\frac{1}{M} \sum_{j=1}^{M} f(z_{ij}, x_i), y_i\right)\right] + \nu G\left(f\right),$$

where $\{(x_i, y_i)\}_{i=1}^N \sim p(x, y)$, and $\{z_{ij}\}_{j=1}^M \sim p(z|x_i)$ for each x_i. To ensure an excess risk of ϵ, both N and M need be at least as large as $\mathcal{O}(1/\epsilon^2)$, making the overall sample required to be $\mathcal{O}(1/\epsilon^4)$; see [70, 136] and references therein. Hence, when M is small, SAA would provide poor results.

A second option is to resort to stochastic gradient methods (SGD). One can construct a *biased* stochastic estimate of the gradient using $\nabla_f L = \nabla \ell(\langle f, \tilde{\psi}(x)\rangle, y)\tilde{\psi}(x) + \nu \nabla_f G(f)$, where $\tilde{\psi}(x)$ is an estimate of $\mathbb{E}_{z|x}[\psi(z, x)]$ for any x. To ensure convergence, the bias of the stochastic gradient must be small, *i.e.*, a large amount of samples from the conditional distribution is needed.

Another commonly used approach is to first represent the conditional distributions as the so-called kernel conditional embedding, and then perform a supervised learning step on the embedded conditional distributions [137, 138]. This two-step procedure suffers from poor statistical sample complexity and computational cost. The kernel conditional embedding estimation costs $O(N^3)$, where N is number of pair of samples (x, z). To achieve ϵ error in the conditional kernel embedding estimation, N needs to be $\mathcal{O}(1/\epsilon^4)$[1].

We in Chapter 3 and [136] independently solved a related but easier problem of the form,

$$\min_{f \in \mathcal{F}} \; L(f) = \mathbb{E}_y \left[\ell(\mathbb{E}_z[f(z)], y)\right] + \nu G(f). \tag{5.8}$$

Despite the different objectives, the algorithm proposed in [136] cannot directly handle random variable z with *infinite support*. Hence, such an algorithm does not apply to the more general and difficult situation that we consider in this chapter.

5.1.2 Contributions

To address the above challenges, we propose a novel approach called *dual embedding*. The key idea is to reformulate (5.1) into a min-max or saddle point problem by utilizing the Fenchel duality of the loss function. We observe that with smooth loss function and continuous conditional distributions, the dual variables form a continuous function of x and y. Therefore, we can parameterize it as a function in some RKHS induced by any universal

[1] With appropriate assumptions on joint distribution $p(x, z)$, a better rate can be obtained [138]. However, for fair comparison, we did not introduce such extra assumptions.

kernel, where the information about the marginal distribution $p(x)$ and conditional distribution $p(z|x)$ can be aggregated via a kernel embedding of the joint distribution $p(x, z)$. Furthermore, we propose an efficient algorithm based on stochastic approximation to solve the resulting saddle point problem over RKHS spaces, and establish finite-sample analysis of the generic learning over dynamics problems.

Compared to previous applicable approaches, an advantage of the proposed method is that it requires only *one sample* from each conditional distribution. Under mild conditions, the overall sample complexity reduces to $\mathcal{O}(1/\epsilon^2)$ in contrast to the $\mathcal{O}(1/\epsilon^4)$ complexity required by SAA or kernel conditional embedding. As a by-product, even in the degenerate case (5.8), this implies an $\mathcal{O}(1/\epsilon^2)$ sample complexity when inner function is linear, which already surpasses the result obtained in [136] and is known to be unimprovable. Furthermore, our algorithm is generic for the family of problems of learning with dynamics, and can be adapted to problems with different loss functions and hypothesis function spaces.

Our proposed method also offers some new insights into several related applications. In reinforcement learning settings, our method provides the first algorithm that truly minimizes the mean-square Bellman error (MSBE) with both theoretical guarantees and sample efficiency. We show that the existing gradient-TD2 algorithm by [139, 140], is a special case of our algorithm, and the residual gradient algorithm [28] is derived by optimizing an upper bound of MSBE. In the invariance learning setting, our method also provides a unified view of several existing methods for encoding invariance. Finally, numerical experiments on both synthetic and real-world datasets show that our method can significantly improve over the previous state-of-the-art performances.

5.2 Preliminaries

We first introduce our notations on Fenchel duality, kernel and kernel embedding. Let $\mathcal{X} \subset \mathbb{R}^d$ be some input space and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel function. For notation simplicity, we denote the feature map of kernel k or $\tilde{k}$ as

$$\phi(x) := k(x, \cdot), \quad \psi(z) := \tilde{k}(z, \cdot),$$

and use $k(x,\cdot)$ and $\phi(x)$, or $\tilde{k}(z,\cdot)$ and $\psi(z)$ interchangeably. We denote all continuous functions on $\mathcal{X}$ as $\mathcal{C}(\mathcal{X})$ and $\|\cdot\|_\infty$ as the maximum norm. We call k a *universal kernel* if $\mathcal{H}$ is dense in $\mathcal{C}(\Omega')$ for any compact set $\Omega' \subseteq \mathcal{X}$, *i.e.*, for any $\epsilon > 0$ and $u \in \mathcal{C}(\Omega')$, there exists $h \in \mathcal{H}$, such that $\|u - h\|_\infty \leqslant \epsilon$. Examples of universal kernel include the Gaussian kernel, $k(x,x') = \exp\left(-\frac{\|x-x'\|_2^2}{\sigma^{-2}}\right)$, Laplacian kernel, $k(x,x') = \exp\left(-\frac{\|x-x'\|_1}{\sigma^{-1}}\right)$, and so on.

Convex conjugate and Fenchel duality Let $\ell : \mathbb{R}^d \to \mathbb{R}$, its convex conjugate function is defined as

$$\ell^*(u) = \sup_{v \in \mathbb{R}^d} \{u^\top v - \ell(v)\}.$$

When ℓ is proper, convex and lower semicontinuous for any u, its conjugate function is also proper, convex and lower semicontinuous. More improtantly, the (ℓ, ℓ^*) are dual to each other, *i.e.*, $(\ell^*)^* = \ell$, which is known as Fenchel duality [141, 142]. Therefore, we can represent the ℓ by its convex conjugate as ,

$$\ell(v) = \sup_{u \in \mathbb{R}^d} \{v^\top u - \ell^*(u)\}.$$

It can be shown that the supremum achieves if $v \in \partial \ell^*(u)$, or equivalently $u \in \partial \ell(v)$.

Function approximation using RKHS Let $\mathcal{H}^\delta := \{h \in \mathcal{H} : \|h\|_{\mathcal{H}}^2 \leqslant \delta\}$ be a bounded ball in the RKHS, and we define the approximation error of the RKHS $\mathcal{H}^\delta$ as the error from approximating continuous functions in $\mathcal{C}(\mathcal{X})$ by a function $h \in \mathcal{H}^\delta$, *i.e.*, [25, 53]

$$\mathcal{E}(\delta) := \sup_{u \in \mathbb{C}(\mathcal{X})} \inf_{h \in \mathcal{H}^\delta} \|u - h\|_\infty. \tag{5.9}$$

One can immediately see that $\mathcal{E}(\delta)$ decreases as δ increases and vanishes to zero as δ goes to infinity. If $\mathcal{C}(\mathcal{X})$ is restricted to the set of uniformly bounded continuous functions, then $\mathcal{E}(\delta)$ is also bounded. The approximation property, *i.e.*, dependence on δ remains an open question for general RKHS, but has been carefully established for special kernels. For example, with the kernel $k(x,x') = 1/(1+\exp(\langle x, x'\rangle))$ induced by the sigmoidal activation function, we have $\mathcal{E}(\delta) = O(\delta^{-2/(d+1)}\log(\delta))$ for Lipschitz continuous function space $\mathcal{C}(\mathcal{X})$ [25].[2]

[2]The rate is also known to be unimprovable by [143].

Hilbert space embedding of distributions Hilbert space embeddings of distributions [47] are mappings of distributions into potentially *infinite* dimensional feature spaces,

$$\mu_x := \mathbb{E}_x[\phi(x)] = \int_{\mathcal{X}} \phi(x)p(x)dx \ : \ \mathcal{P} \mapsto \mathcal{H} \tag{5.10}$$

where the distribution is mapped to its expected feature map, *i.e.*, to a point in a feature space. Kernel embedding of distributions has rich representational power. Some feature map can make the mapping injective [144], meaning that if two distributions are different, they are mapped to two distinct points in the feature space. For instance, when $\mathcal{X} \subseteq \mathbb{R}^d$, the feature spaces of many commonly used kernels, such as the Gaussian RBF kernel, will generate injective embedding. We can also embed the joint distribution $p(x, y)$ over a pair of variables using two kernels $k(x, x) = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$ and $\tilde{k}(z, z') = \langle \psi(z), \psi(z') \rangle_{\mathcal{G}}$ as

$$\begin{aligned} \mathcal{C}_{zx} &:= \mathbb{E}_{zx}[\psi(z) \otimes \phi(x)] \\ &= \int_{\mathcal{Z}\times\mathcal{X}} \psi(z) \otimes \phi(x)p(z, x)dzdx : \mathcal{P} \mapsto \mathcal{H} \otimes \mathcal{G}, \end{aligned}$$

where the joint distribution is mapped to a point in a tensor product feature space. Based on embedding of joint distributions, kernel embedding of conditional distributions can be defined as $\mathcal{C}_{z|x} := \mathcal{C}_{zx}\mathcal{C}_{xx}^{-1}$ as an operator $\mathcal{H} \mapsto \mathcal{G}$ [137]. With $\mathcal{C}_{z|x}$, we can obtain the expectations easily, *i.e.*,

$$\mathbb{E}_{z|x}[g(z)] = \langle g, \langle \mathcal{C}_{z|x}, \phi(x) \rangle_{\mathcal{H}} \rangle_{\mathcal{G}}. \tag{5.11}$$

Both the joint distribution embedding, $\mathcal{C}_{zx}$, and the conditional distribution embedding, $\mathcal{C}_{z|x}$, can be estimated from *i.i.d.* samples $\{(x_i, z_i)\}_{i=1}^N$ from $p(x, z)$ or $p(z|x)$, respectively [47, 137], as

$$\widehat{\mathcal{C}}_{zx} = \frac{1}{N}\Psi\Upsilon^\top, \text{ and } \quad \widehat{\mathcal{C}}_{z|x} = \Psi(K + \lambda I)^{-1}\Upsilon^\top,$$

where $\Psi = (\psi(z_1), \ldots, \psi(z_N))$, $\Upsilon = (\phi(x_1), \ldots, \phi(x_N))$, and $K = \Upsilon^\top\Upsilon$. Due to the inverse of $K + \lambda I$, the kernel conditional embedding estimation requires $\mathcal{O}(N^3)$ cost.

5.3 Dual Embedding Framework

In this section, we propose a novel and sample-efficient framework to solve problem (5.1). Our framework leverages Fenchel duality and feature space embedding technique to bypass the difficulties of nested expectation and the need for overwhelmingly large sample from conditional distributions. We start by introducing the interchangeability principle, which plays a fundamental role in our method.

Lemma 20 (interchangeability principle) *Let ξ be a random variable on Ξ and assume for any $\xi \in \Xi$, function $g(\cdot, \xi) : \mathbb{R} \to (-\infty, +\infty)$ is a proper*[3] *and upper semicontinuous*[4] *concave function. Then*

$$\mathbb{E}_\xi[\max_{u\in\mathbb{R}} g(u,\xi)] = \max_{u(\cdot)\in\mathcal{G}(\Xi)} \mathbb{E}_\xi[g(u(\xi),\xi)].$$

where $\mathcal{G}(\Xi) = \{u(\cdot) : \Xi \to \mathbb{R}\}$ is the entire space of functions defined on support Ξ.

The result implies that one can replace the expected value of point-wise optima by the optimum value over a function space. For the proof of lemma 20, please refer to Appendix C.1. More general results of interchange between maximization and integration can be found in [145, Chapter 14] and [146, Chapter 7].

5.3.1 Saddle Point Reformulation

Let the loss function $\ell_y(\cdot) := \ell(\cdot, y)$ in (5.1) be a proper, convex and lower semicontinuous for any y. For the clarity, we set $\nu = 0$ in the following discussion. However, it can easily generalized to the case $\nu > 0$ without any effect. We denote $\ell_y^*(\cdot)$ as the convex conjugate; hence $\ell_y(v) = \max_u\{uv - \ell_y^*(u)\}$, which is also a proper, convex and lower semicontinuous function. Using the Fenchel duality, we can reformulate problem (5.1) as

$$\min_{f\in\mathcal{F}} \mathbb{E}_{xy}\left[\max_{u\in\mathbb{R}}\left[\mathbb{E}_{z|x}[f(z,x)]\cdot u - \ell_y^*(u)\right]\right], \tag{5.12}$$

[3] We say $g(\cdot,\xi)$ is proper when $\{u \in \mathbb{R} : g(u,\xi) < \infty\}$ is non-empty and $g(u,\xi) > -\infty$ for $\forall u$.

[4] We say $g(\cdot,\xi)$ is upper semicontinuous when $\{u \in \mathbb{R} : g(u,\xi) < \alpha\}$ is an open set for $\forall\alpha \in \mathbb{R}$. Similarly, we say $g(\cdot,\xi)$ is lower semicontinuous when $\{u \in \mathbb{R} : g(u,\xi) > \alpha\}$ is an open set for $\forall\alpha \in \mathbb{R}$.

Note that by the concavity and upper-semicontinuity of $-\ell_y^*(\cdot)$, for any given pair (x, y), the corresponding maximizer of the inner function always exists. Based on the interchangeability principle stated in Lemma 20, we can further rewrite (5.12) as

$$\min_{f\in\mathcal{F}} \max_{u(\cdot)\in\mathcal{G}(\Xi)} \Phi(f,u) := \mathbb{E}_{zxy}[f(z,x)\cdot u(x,y)] - \mathbb{E}_{xy}[\ell_y^*(u(x,y))], \tag{5.13}$$

where $\Xi = \mathcal{X}\times\mathcal{Y}$ and $\mathcal{G}(\Xi) = \{u(\cdot) : \Xi \to \mathbb{R}\}$ is the entire function space on Ξ. We emphasize that the max-operator in (5.12) and (5.13) have different meanings: the one in (5.12) is taking over a single variable, while the other one in (5.13) is over all possible function $u(\cdot) \in \mathcal{G}(\Xi)$.

Now that we have eliminated the nested expectation in the problem of interest, and converted it into a stochastic saddle point problem with an additional dual function space to optimize over. By definition, $\Phi(f,u)$ is always concave in u for any fixed f. Since $f(z,x) = \langle f, \psi(z,x)\rangle$, $\Phi(f,u)$ is also convex in f for any fixed u. Our reformulation (5.13) is indeed a convex-concave saddle point problem.

5.3.2 Dual Continuation

Although the reformulation in (5.13) gives us more structure of the problem, it is not yet tractable in general. This is because the dual function $u(\cdot)$ can be an arbitrary function which we do not know how to represent. In the following, we will introduce a tractable representation for (5.13).

First, we will define the function $u^*(\cdot) : \Xi = \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ as the *optimal dual function* if for any pair $(x, y) \in \Xi$,

$$u^*(x,y) \in \operatorname{argmax}_{u\in\mathbb{R}} \left\{u\cdot\mathbb{E}_{z|x}[f(z,x)] - \ell_y^*(u)\right\}.$$

Note the optimal dual function is well-defined since the optimal set is nonempty. Furthermore, $u^*(x,y)$ is related to the conditional distribution via $u^*(x,y) \in \partial\ell_y(\mathbb{E}_{z|x}[f(z,x)])$. This can be simply derived from convexity of loss function and Fenchel's inequality; see [141] for a more formal argument. Depending on the property of the loss function $\ell_y(v)$, we can

further derive that (see proofs in Appendix C.1):

Proposition 21 *Suppose both $f(z,x)$ and $p(z|x)$ are continuous in x for any z,*

(1) (Discrete case) If the loss function $\ell_y(v)$ is continuously differentiable in v for any $y \in \mathcal{Y}$, then $u^*(x,y)$ is unique and continuous in x for any $y \in \mathcal{Y}$;

(2) (Continuous case) If the loss function $\ell_y(v)$ is continuously differentiable in (v,y), then $u^*(x,y)$ is unique and continuous in (x,y) on $\mathcal{X} \times \mathcal{Y}$.

This assumption is satisfied widely in real-world applications. For instance, when it comes to the policy evaluation problem in 5.3, the corresponding optimal dual function is continuous as long as the reward function is continuous, which is true for many reinforcement learning tasks.

The fact that the optimal dual function is a continuous function has interesting consequences. As we mentioned earlier, the space of dual functions can be arbitrary and difficult to represent. Now we can simply restrict the parametrization to the space of continuous functions, which is tractable and still contains the global optimum of the optimization problem in (5.13). This also provides us the basis for using an RKHS to approximate these dual functions, and simply optimizing over the RKHS.

5.3.3 Feature Space Embedding

In the rest of this chapter, we assume conditions described in Proposition 21 always hold. For the sake of simplicity, we focus only on the case when $\mathcal{Y}$ is a continuous set. Hence, from Proposition 21, the optimal dual function is indeed continuous in $(x,y) \in \Xi = \mathcal{X} \times \mathcal{Y}$. As an immediate consequence, we lose nothing by restricting the dual function space $\mathcal{G}(\Xi)$ to be continuous function space on Ξ. Recall that with the universal kernel, we can approximate any continuous function with arbitrarily small error. Thus we approximate the dual space $\mathcal{G}(\Xi)$ by the bounded RKHS $\mathcal{H}^\delta$ induced by a universal kernel $k((x,y),(x',y')) = \langle \phi(x,y), \phi(x',y') \rangle_{\mathcal{H}}$ where $\phi(\cdot)$ is the implicit feature map, *i.e.*, $u(x,y) = \langle u, \phi(x,y) \rangle_{\mathcal{H}}$. The universal kernel on $\Xi = \mathcal{X} \times \mathcal{Y}$ can be easily obtained by $k((x,y),(x',y')) := k_{\mathcal{X}}(x,x') \otimes k_{\mathcal{Y}}(y,y')$ as long as $k_{\mathcal{X}}(x,x')$ and $k_{\mathcal{Y}}(y,y')$ are universal and $\mathcal{X}$ and $\mathcal{Y}$ are local compact

Polish spaces[5], as proved in Theorem 5 in [147]. Note that $\mathcal{H}^\delta$ is a subspace of the continuous function space, and hence is a subspace of the dual space $\mathcal{G}(\Xi)$.

To distinguish inner product between the primal function space $\mathcal{F}$ and the dual RKHS $\mathcal{H}^\delta$, we denote the inner product in $\mathcal{F}$ as $\langle \cdot, \cdot \rangle_{\mathcal{F}}$. We can rewrite the saddle point problem in (5.13) as

$$\begin{aligned} \min_{f \in \mathcal{F}} \max_{u \in \mathcal{H}^\delta} \Phi(f, u) &= \mathbb{E}_{xyz} \left[\langle f, \psi(z, x) \rangle_{\mathcal{F}} \cdot \langle u, \phi(x, y) \rangle_{\mathcal{H}} - \ell_y^*(\langle u, \phi(x, y) \rangle_{\mathcal{H}}) \right] \\ &= f^\top \mathcal{C}_{zxy} u - \mathbb{E}_{xy}[\ell_y^*(\langle u, \phi(x, y) \rangle_{\mathcal{H}})], \end{aligned} \tag{5.14}$$

where $f(z, x) = \langle f, \psi(z, x) \rangle_{\mathcal{F}}$ by the definition of $\mathcal{F}$, and $\mathcal{C}_{zxy} = \mathbb{E}_{zxy}[\psi(z, x) \otimes \phi(x, y)]$ is the joint embedding of $p(z, x, y)$ over $\mathcal{F} \times \mathcal{H}$. The new saddle point approximation (5.14) based on dual kernel embedding allows us to efficient represent the dual function and get away from the fundamental difficulty with insufficient sampling from the conditional distribution. There is no need to access either the conditional distribution $p(z|x)$, the conditional expectation $\mathbb{E}_{z|x}[\cdot]$, or the conditional embedding operator $\mathcal{C}_{z|x}$ anymore, therefore, reducing both the statistical and computational complexity.

Specifically, given a pair of sample (x, y, z), where $(x, y) \sim p(x, y)$ and $z \sim p(z|x)$, we can now easily construct an unbiased stochastic estimate for the gradient, namely,

$$\begin{aligned} \nabla_f \hat{\Phi}_{x,y,z}(f, u) &= \psi(z, x) u(x, y), \\ \nabla_u \hat{\Phi}_{x,y,z}(f, u) &= [f(z, x) - \nabla \ell_y^*(u(x, y))] \phi(x, y), \end{aligned}$$

with $\mathbb{E}\left[\nabla \hat{\Phi}_{x,y,z}(f, u)\right] = \nabla \Phi(f, u)$, respectively. For simplicity of notation, we use ∇ to denote the subgradient as well as the gradient. With the unbiased stochastic gradient, we are now able to solve the approximation problem (5.14) by resorting to the powerful mirror descent stochastic approximation framework [70].

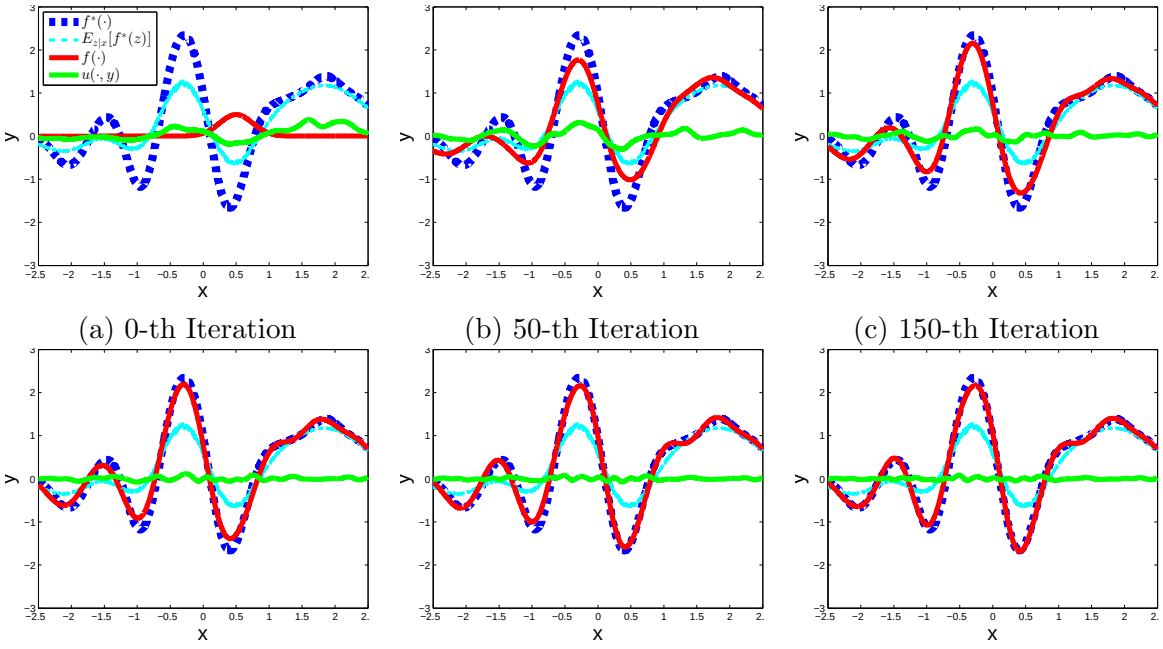
Put every component together, the algorithm is summarized in Algorithm 4. At each iteration, the algorithm performs a projected gradient step both for the primal variable f and dual variable u based on the unbiased stochastic gradient. The proposed algorithm

[5] A topological space is called Polish if it is complete, separable and metrizable, *e.g.*, $\mathbb{R}^d$.

Algorithm 4 Embedding-SGD for Optimization (5.14)

Input: $p(x,y)$, $p(z|x)$, $\psi(z,x)$, $\phi(x,y)$, $\{\gamma_i \geqslant 0\}_{i=1}^t$

1: **for** $i = 1, \ldots, t$ **do**
2: Sample $(x_i, y_i) \sim p(x,y)$ and $z_i \sim p(z|x)$.
3: $f_{i+1} = \Pi_{\mathcal{F}}(f_i - \gamma_i \psi(z_i, x_i) u_i(x_i, y_i))$.
4: $u_{i+1} = \Pi_{\mathcal{H}^\delta}(u_i + \gamma_i [f_i(z_i, x_i) - \nabla \ell_{y_i}^*(u_i(x_i, y_i))]\phi(x_i, y_i))$
5: **end for**
Output: $\bar{f}_t = \frac{\sum_{i=1}^t \gamma_i f_i}{\sum_{i=1}^t \gamma_i}$, $\bar{u}_t = \frac{\sum_{i=1}^t \gamma_i u_i}{\sum_{i=1}^t \gamma_i}$



(a) 0-th Iteration (b) 50-th Iteration (c) 150-th Iteration

(a) 400-th Iteration (b) 1000-th Iteration (c) 2000-th Iteration

Figure 5.1: Toy example with f^* sampled from a Gaussian processes. The y at position x is obtained by smoothing f^* with a Gaussian distribution condition on location x, *i.e.*, $y = \mathbb{E}_{z|x}[f^*(z)]$ where $z \sim p(z|x) = \mathcal{N}(x, 0.3)$. Given samples $\{x, y\}$, the task is to recover $f^*(\cdot)$. The blue dash curve is the ground-truth $f^*(\cdot)$. The cyan curve is the observed noisy y. The red curve is the recovered signal $f(\cdot)$ and the green curve denotes the dual function $u(\cdot, y)$ with the observed y plugged for each corresponding position x. Indeed, the dual function $u(\cdot, y)$ emphasizes the difference between y and $\mathbb{E}_{z|x}[f(z)]$ on every x. The interaction between primal $f(\cdot)$ and dual $u(\cdot, y)$ results in the recovery of the denoised signal.

avoids the need for overwhelmingly large sample sizes from the conditional distributions when estimating the gradient. At each iteration, only one sample from the conditional distribution is required in our algorithm!

An example Let us illustrate this through a concrete example. Let $f^*(\cdot) \in \mathcal{F}$ be the true function, and output $y = \mathbb{E}_{z|x}[f^*(z)]$ given x. We can recover the true function $f^*(\cdot)$ by solving the optimization problem

$$\min_{f \in \mathcal{F}} \mathbb{E}_{xy}\left[\frac{1}{2}\left(y - \mathbb{E}_{z|x}[f(z)]\right)^2\right].$$

In this example, $\ell_y(v) = \frac{1}{2}(y-v)^2$ and $\ell_y^*(u) = uy + \frac{1}{2}u^2$. Invoking the saddle point reformulation, this leads to

$$\min_{f \in \mathcal{F}} \max_{u \in \mathcal{H}^\delta(\Xi)} \mathbb{E}_{xyz}\left[\left(f(z) - y\right) u(x,y)\right] - \frac{1}{2}\mathbb{E}_{xy}\left[u(x,y)^2\right],$$

where the dual function $u(x,y)$ fits the discrepancy between y and $\mathbb{E}_{z|x}[f(z)]$, and thus, promotes the performance of primal function by emphasizing the different positions. See Figure 5.1 for the illustration of the interaction between the primal and dual functions in the proposed Embedding-SGD algorithm.

5.4 Theoretical Analysis

In this section, we provide the convergence rate and sample complexity of the embedding-SGD algorithm. Throughout our discussion, we make the following standard assumptions:

Assumption 8 *There exists constant scalars $C_{\mathcal{F}}$, $M_{\mathcal{F}}$, and c_ℓ, such that for any $f \in \mathcal{F}, u \in \mathcal{H}^\delta$,*

$$\mathbb{E}_{z,x}[\|f(z,x)\|_2^2] \leqslant M_{\mathcal{F}}, \quad \mathbb{E}_{z,x}[\|\psi(z,x)\|_{\mathcal{F}}^2] \leqslant C_{\mathcal{F}}, \quad \mathbb{E}_y[\|\nabla \ell_y^*(u)\|_2^2] \leqslant c_\ell.$$

Assumption 9 *There exists constant $\kappa > 0$ such that $k(w,w') \leqslant \kappa$ for any $w, w' \in \mathcal{X}$.*

Assumption 8 and 9 basically suggest that the variance of our stochastic gradient estimate is always bounded. Note that we do not assume any strongly convexity or concavity of the saddle point problem, or Lipschitz smoothness. Hence, we set the output as the average of intermediate solutions weighted by the learning rates $\{\gamma_i\}$, as often used in the literature,

to ensure the convergence of the algorithm.

Define the accuracy of any candidate solution $(\bar{f}, \bar{u})$ to the saddle point problem as

$$\epsilon_{\text{gap}}(\bar{f}, \bar{u}) := \max_{u \in \mathcal{H}^\delta} \Phi(\bar{f}, u) - \min_{f \in \mathcal{F}} \Phi(f, \bar{u}). \tag{5.15}$$

We have the following convergence result,

Theorem 22 *Under Assumptions 8 and 9, the solution $(\bar{f}_t, \bar{u}_t)$ after t steps of the algorithm with step-sizes being $\gamma_t = \frac{\gamma}{\sqrt{t}} (\gamma > 0)$ satisfies:*

$$\mathbb{E}[\epsilon_{\text{gap}}(\bar{f}_t, \bar{u}_t)] \leqslant [(2D_{\mathcal{F}}^2 + 4\delta)/\gamma + \gamma \mathcal{C}(\delta, \kappa)] \frac{1}{\sqrt{t}} \tag{5.16}$$

where $D_{\mathcal{F}}^2 = \sup_{f \in \mathcal{F}} \frac{1}{2} \|f_0 - f\|_2^2$ and $\mathcal{C}(\delta, \kappa) = \kappa(5M_{\mathcal{F}} + c_\ell) + \frac{1}{8}(\delta + \kappa)^2 C_{\mathcal{F}}$.

The above theorem implies that our algorithm achieves an overall $\mathcal{O}(1/\sqrt{t})$ convergence rate, which is known to be unimprovable already for traditional stochastic optimization with general convex loss function [70]. We further observe that

Proposition 23 *If $f(z, x)$ is uniformly bounded by C and $\ell_y^*(v)$ is uniformly K-Lipschitz continuous in v for any y, then $\Phi(f, u)$ is $(C+K)$-Lipschitz continuous on $\mathcal{G}(\Xi)$ with respect to $\|\cdot\|_\infty$, i.e.*

$$|\Phi(f, u_1) - \Phi(f, u_2)| \leqslant (C + K)\|u_1 - u_2\|_\infty, \forall u_1, u_2 \in \mathcal{G}(\Xi).$$

Let f_* be the optimal solution to (5.1). Invoking the Lipschitz continuity of Φ and using standard arguments of decomposing the objective, we have

$$L(\bar{f}_t) - L(f_*) \leqslant \epsilon_{\text{gap}}(\bar{f}_t, \bar{u}_t) + 2(C + K)\mathcal{E}(\delta).$$

Combining Proposition 23 and Theorem 22, we finally conclude that under the conditions therein,

$$\mathbb{E}[L(\bar{f}_t) - L(f_*)] \leqslant \mathcal{O}\left(\frac{\delta^{3/2}}{\sqrt{t}} + \mathcal{E}(\delta)\right). \tag{5.17}$$

There is clearly a delicate trade-off between the optimization error and approximation error. Using large δ will increase the optimization error but decrease the approximation error. When δ is moderately large (which is expected in the situation when the optimal dual function has small magnitude), our dual embedding algorithm can achieve an overall $\mathcal{O}(1/\epsilon^2)$ sample complexity when solving learning problems in the form of (5.1). For the analysis details, please refer to Appendix C.3.

5.5 Practical Applications

In this section, we discuss in details how the dual kernel embedding can be applied to solve several important learning problems in machine learning, *e.g.*, learning with invariance and reinforcement learning, which are the special cases of the optimization (5.1). By simple verification, these examples satisify our assumptions for the convergence of algorithm. We tailor the proposed algorithm for the respective learning scenarios and unify several existing algorithms for each learning problem into our framework. We only focus on algorithms with kernel embedding. Extended algorithms with random feature, doubly SGD, neural networks as well as their hybrids can be found in Appendix C.5.1, C.5.2 and C.5.3.

5.5.1 Learning with Invariant Representations

Invariance learning The goal is to solve the optimization (5.2), which learns a function in RKHS $\tilde{\mathcal{H}}$ with kernel $\tilde{k}$. Applying the dual kernel embedding, we end up solving the saddle point problem

$$\min_{f\in\tilde{\mathcal{H}}}\max_{u\in\mathcal{H}}\ \mathbb{E}_{zxy}\left[\langle f,\psi(z)\rangle_{\tilde{\mathcal{H}}}\cdot u(x,y)\right]-\mathbb{E}_{xy}[\ell_y^*(u(x,y))]+\frac{\nu}{2}\|f\|_{\tilde{\mathcal{H}}}^2,$$

where $\mathcal{H}$ is the dual RKHS with the universal kernel introduced in our method.

Remark: The proposed algorithm bears some similarities to virtual sample techniques [134, 148] in the sense that they both create examples with prior knowledge to incorporate invariance. In fact, the virtual sample technique can be viewed as optimizing an upper bound of the objective (5.2) by simply moving the conditional expectation outside, *i.e.*, $\mathbb{E}_{x,y}[\ell(y,\mathbb{E}_{z|x}[f(z)])]\leqslant\mathbb{E}_{x,y,z}\big[\ell(y,f(z))\big]$, where the inequality comes from convexity of

$\ell(y, \cdot)$.

Remark: The learning problem (5.2) can be understood as learning with RKHS $\hat{\mathcal{H}}$ with Haar-Integral kernel $\hat{k}$ which is generated by $\tilde{k}$ as

$$\hat{k}(x, x') = \langle \mathbb{E}_{p(z|x)}[\psi(z)], \mathbb{E}_{p(z'|x')}[\psi(z')] \rangle_{\tilde{\mathcal{H}}},$$

with implicit feature map $\mathbb{E}_{p(z|x)}[\psi(z)]$. If $f \in \tilde{\mathcal{H}}$, then, $f(x) = \mathbb{E}_{z|x}[\langle f, \psi(z) \rangle_{\tilde{\mathcal{H}}}] = \langle f, \mathbb{E}_{z|x}[\psi(z)] \rangle \in \hat{\mathcal{H}}$. The Haar-Integral kernel can be viewed as a special case of Hilbertian metric on probability measures on which the output of function should be invariant [149]. Therefore, other kernels defined for distributions, *e.g.*, the probability product kernel [150], can be also used in incorporating invariance.

Remark: Robust learning with contamined samples can also be viewed as incorporating invariance prior with respect to the perturbation distribution into learning procedure. Therefore, rather than resorting to robust optimization techniques [151, 152], the proposed algorithm for learning with invariance serves as a viable alternative for robust learning.

5.5.2 Policy Evaluation in Reinforcement Learning

Policy evaluation The goal is to estimate the value function $V^\pi(\cdot)$ of a given policy $\pi(a|s)$ by minimizing the mean-square Bellman error (MSBE) (5.3). With $V^\pi \in \tilde{\mathcal{H}}$ with feature map $\psi(\cdot)$, we apply the dual kernel embedding, which will lead to the saddle point problem

$$\min_{V^\pi \in \tilde{\mathcal{H}}} \max_{u \in \mathcal{H}} \quad \mathbb{E}_{s',a,s} \left[\left(R(s,a) - \langle V^\pi, \psi(s) - \gamma \psi(s') \rangle_{\tilde{\mathcal{H}}} \right) u(s) \right] - \frac{1}{2} \mathbb{E}_s[u^2(s)]. \quad (5.18)$$

Remark: The algorithm can be extended to off-policy setting. Let π_b be the behavior policy and $\varpi(a|s) = \frac{\pi(a|s)}{\pi_b(a|s)}$ be the importance weight, then the objctive will be adjusted by $\varpi(a|s)$, *i.e.*

$$\min_{V^\pi \in \tilde{\mathcal{H}}} \mathbb{E}_s \left[\left(\mathbb{E}_{s',a|s} \left[\varpi(a|s) \left(R(s,a) - \langle V^\pi, \psi(s) - \gamma \psi(s') \rangle_{\tilde{\mathcal{H}}} \right) \right] \right)^2 \right]$$

where the successor state $s' \sim P(s'|s,a)$ and actions $a \sim \pi_b(a|s)$ from behavior policy. With

the dual kernel embedding, we can derive similar algorithm for off-policy setting, with extra importance weight $\varpi(a|s)$ to adjust the sample distribution.

Remark: We used different RKHSs for primal and dual functions. If we use the *same finite basis functions* to parametrize both the value function and the dual function, *i.e.*, $V^\pi(s) = \theta^T\psi(s)$ and $u(s) = \eta^T\psi(s)$, where $\psi(s) = [\psi_i(s)]_{i=1}^d \in \mathbb{R}^d$, $\theta, \eta \in \mathbb{R}^d$, our saddle point problem (5.18) reduces to $\min_\theta \left\|\mathbb{E}_{s,a,s'}[\Delta_\theta(s,a,s')\psi]\right\|^2_{\mathbb{E}[\psi\psi^\top]^{-1}}$, where $\Delta_\theta(s,a,s') = R(s,a) + \gamma V^\pi(s') - V^\pi(s)$. This is exactly the same as the objective proposed in [139] of gradient-TD2. Moreover, the update rules in gradient-TD2 can also be derived by conducting the proposed Embedding-SGD with such parametrizations. For details of the derivation, please refer to Appendix C.4.

From this perspective, gradient-TD2 is simply a special case of the proposed Embedding-SGD applied to policy evaluation with particular parametrization. However, in the view of our framework, there is really no need to, and should not, restrict to the same finite parametric model for the value and dual functions. As further demonstrated in our experiments, with different nonparametric models, the performances can be improved significantly. See details in Section 5.6.2.

The residual gradient (RG) [28] is trying to apply stochastic gradient descent directly to the MSBE with finite parametric form of value function, *i.e.*, $V^\pi(s) = \theta^T\psi(s)$, resulting the gradient as

$$\mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')\psi(s)\right] - \gamma\mathbb{E}_s\left[\mathbb{E}_{s',a|s}\left[\Delta_\theta(s,a,s')\right]\mathbb{E}_{s'|s}\left[\psi(s')\right]\right].$$

Due to the inside conditional expectation in gradient expression, to obtain an unbiased estimator of the gradient, it requires two independent samples of s' given s, which is not practical. To avoid the double-sampling issue, [28] suggests to use an estimator of the gradient as $\mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')\left(\psi(s) - \gamma\psi(s')\right)\right]$. Then, the algorithm is actually optimizing $\mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')^2\right]$, which is an upper bound of MSBE (5.3) because of the convexity of square loss.

Our algorithm is also fundamentally different from the TD algorithm even in the finite state case. The TD algorithm updates the state-value function directly by an estimate of

the temporal difference based on one pair of samples, while our algorithm updates the state-value function based on accumulated estimate of the temporal difference, which intuitively is more robust.

Optimal control The goal is to estiamte the $z(\cdot)$ function by minimizing the error in linear Bellamn equation (5.5). With $z \in \tilde{\mathcal{H}}$ with feature map $\psi(\cdot)$, we apply the dual kernel embedding, which will lead to the saddle point problem

$$\min_{z\in\tilde{\mathcal{H}}} \max_{u\in\mathcal{H}} \Phi(z,u) := \mathbb{E}_{s',s}\left[\langle z, \psi(s) - \exp(-R(s))\psi(s')\rangle_{\tilde{\mathcal{H}}} \cdot u(s)\right] - \frac{1}{2}\mathbb{E}_s[u^2(s)]. \tag{5.19}$$

With the learned z^*, we can recover the optimal control via its conditional distribution $p^{\pi_*}(s'|s) = \frac{p(s'|s)z^*(s)}{\mathbb{E}_{s'|s\sim p(s'|s)}[z^*(s')]}$.

5.5.3 Events Prediction in Stochastic Processes

Expected Hitting Time The goal is to estimate the expected hitting time $h(\cdot,\cdot)$ which minimizes the recursive error (5.7). With $h \in \tilde{\mathcal{H}}$ with feature map $\psi(\cdot,\cdot)$, we apply the dual kernel embedding similarly to the policy evaluation, which will lead to the saddle point problem

$$\min_{h\in\tilde{\mathcal{H}}} \max_{u\in\mathcal{H}} \Phi(h,u) := \mathbb{E}_{s,t}\mathbb{E}_{s'|s,t}\left[\left(1 - \langle h, \psi(s,t) - \psi(s',t)\rangle_{\tilde{\mathcal{H}}}\right) u(s,t)\right] - \frac{1}{2}\mathbb{E}_s\left[u(s,t)^2\right]. \tag{5.20}$$

Remark: With the proposed algorithm, we can estimate $h(s,t)$, the expected hitting time starting from state s and hitting state t, even without observing the hitting events actually happen. We only need to collect the trajectory of the stochastic processes, or just the one-step transition, to feed to the algorithm, therefore, utilize the data more efficietly.

5.6 Experiments

We test the proposed algorithm for two applications, *i.e.*, learning with invariant representation and policy evaluation. In all experiemnts, we conduct comparison on algorithms to optimize the objective with regularization on both primal and dual functions. Since the

target is evaluating the performance of algorithms on the same problem, we fix the weights of the regularization term for the proposed algorithm and the competitors for fairness. The other paramters of models and algorithms, *e.g.*, step size, mini-batch size, kernel parameters and so on, are set according to different tasks.

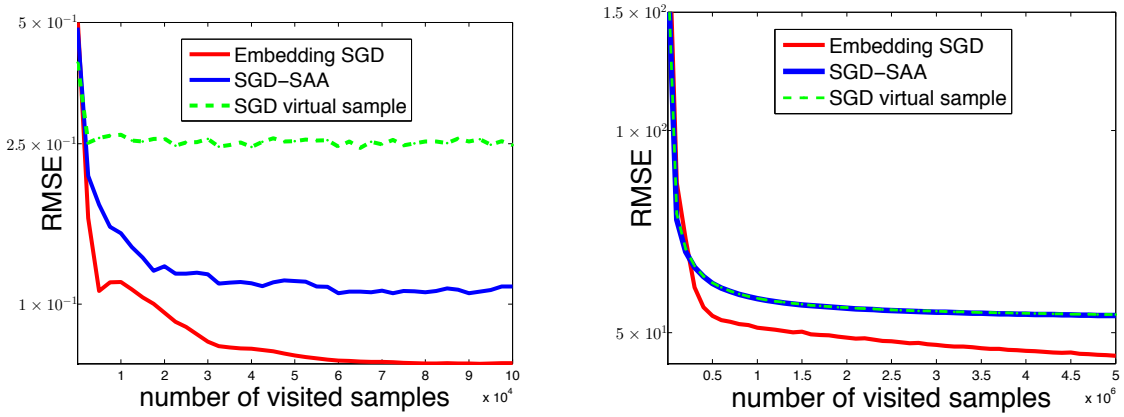
5.6.1 Invariance Learning

To justify the algorithm for learning with invariance, we test the algorithm on two tasks. We first apply the algorithm to robust learning problem where the inputs are contaminated, and then, we conduct comparison on molecular eneretics prediction problem [86]. We compare the proposed algorithm with SGD with virtual samples technique [134, 148] and SGD with finite sample average for inner expectation (SGD-SAA). We use Gaussian kernel in all tasks. To demonstrate the sample-efficiency of our algorithm, 10 virtual samples are generated for each datum in training phase. The algorithms are terminated after going through 10 rounds of training data. We emphasize that the SGD with virtual samples is optimizing an upper bound of the objective, and thus, it is predictable that our algorithm can achieve better performance. We plot its result with dot line instead.

Noisy measurement We generate a synthetic dataset by

$$\begin{aligned} \bar{x} &\sim \mathcal{U}nif([-0.5, 0.5]), \quad x = \bar{x} + 0.05e, \\ y &= (\sin(3.53\pi\bar{x}) + \cos(7.7\pi\bar{x}))\exp(-1.6\pi|\bar{x}|) + 3\bar{x}^2 + 0.01e, \end{aligned}$$

where the contamination $e \sim \mathcal{N}(0, 1)$. Only (x, y) are provided to learning methods, while $\bar{x}$ is unknown. We select the best $\eta \in \{0.1, 1, 10\}$ and $n_0 \in \{1, 10, 100\}$. We use Gaussian kernel for both primal and dual function, whose bandwidth σ are selected from $\{0.05, 0.1, 0.15, 0.2\}$. We set the batch size to be 50. The virtual samples are sampled from $z \sim \mathcal{N}(x, 0.05^2)$ for each observation. In testing phase, the observation is noisyless. The 10 runs average results are illustrated in Figure 5.2(a). The proposed algorithm achieves average MSE as low as 0.0029 after visit 0.1M data, significantly better than the alternatives.



(a) Robust Learning (b) QuantumMachine

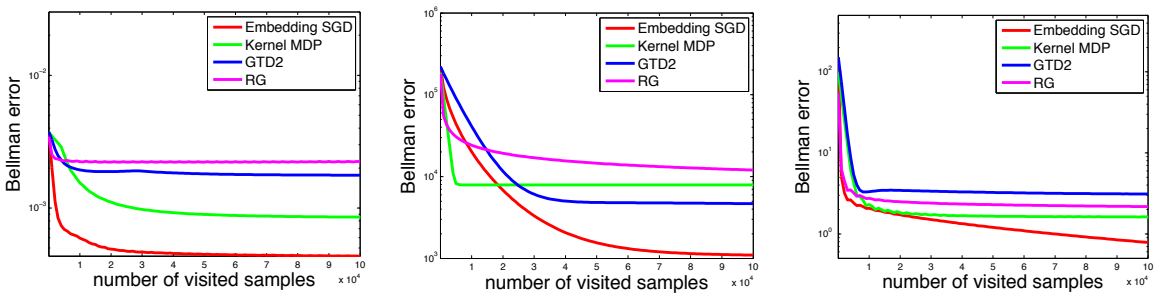
Figure 5.2: Empirical comparison of the Embedding-SGD on learning with invariance task.

QuantumMachine We test the proposed algorithm for learning with invariance task on QuantumMachine 5-fold dataset for atomization energy prediction. We selected the stepsize parameters $\eta \in \{0.1, 0.5, 1\}$ and $n_0 \in \{100, 1000\}$. We adopted Gaussian kernel whose bandwidth is selected by median trick with coeffcient in $\{0.1, 0.25, 0.5, 1\}$. The batch size is set to be 1000. We follow [86] that the data points are represented by Coulomb matrices, and the virtual samples are generated by random permutation. To illustrate the benefits of sample efficiency, we generated 10 virtual samples in training phase and 20 in testing phase. Notice that the number of virtual samples in this experiment is fewer than the one used in [11], therefore, the results are not directly comparable.

The average results are shown in Figure 5.2(b). The proposed algorithm achieves a significant better solution, while SGD-SAA and SGD with virtual samples stuck in inferior solutions due to the inaccurate inner expectation estimation and optimizing indirect objective, respectively.

5.6.2 Policy Evaluation

We compare the proposed algorithm to several prevailing algorithms for policy evaluation, including gradient-TD2 (GTD2) [139, 140], residual gradient (RG) [28] and kernel MDP [153] in terms of mean square Bellman error [154]. It should point out that kernel MDP is not an online algorithm, since it requires to visit the entire dataset when estimating the embedding and inner expectation in each iteration. We conduct experiments for policy



(a) Navigation (b) Cart-Pole (c) PUMA-560

Figure 5.3: Empirical comparison of the Embedding-SGD on policy evaluation on several benchmarks.

evaluation on several benchmark datasets, including navigation, cart-pole swing up and PUMA-560 manipulation. We use Gaussian kernel in the nonparametric algorithms, *i.e.*, kernel MDP and Embedding SGD, while we test random Fourier features [62] for the parametric competitors, *i.e.*, GTD2 and RG. In order to demonstrate the sample efficiency of our method, we only use one sample from the conditional distribution in the training phase, therefore, cross-validation based on Bellman error is not appropriate. We perform a parameter sweep to select the hyper-parameters as [155]. We evaluated all the algorithms in terms of mean square Bellman error on the testing states. On each state s, the mean square Bellman error is estimated with 100 next states s' samples. We set the number of the basis functions in GTD2 and RG to be 2^8. To achieve the convergence property based the theorem [70], we set stepsize to be be $\frac{\eta}{n_0+\sqrt{t}}$ in the proposed algorithm and GTD2, $\frac{\eta}{n_0+t}$ in Kernel MDP and RG. Results are averaged over 10 independent trials.

Navigation The navigation in an unbounded room task extending the discretized MDP in [153] to continuous-state continuous-action MDP. Specifically, the reward is $R(s) = \exp(-100\|s\|^2)$ centered in the middle of the room and $s \sim \mathcal{N}(0, 0.2I)$. We evaluate the deterministic policy policy $\pi(s) = -0.2sR(s)$, following the gradient of the reward function. The transition distribution follows Gaussian distribution, $p(s'|a, s) = \mathcal{N}(s + a, 0.1I)$. The batch size is set to be 20. $\{\eta, n_0\} \in \{0.1, 1, 10\}$. We adopted Gaussian kernel and select the best primal and dual kernel bandwidth in range $\{0.01, 0.05, 0.1, 0.15, 0.2\}$. The γ in MDP is set to be 0.9. Results are reported in Figure 5.3(a).

Cart-pole swing up The cart-pole system consists of a cart and a pendulum. It is an under-actuated system with only one control act on the cart. The goal is to swing-up the pendulum from the initial position (point down). The reward is $R(s) = \frac{1}{2}(s_1^2 + s_2^2 + s_3^2 + 5(s_4 - \pi)^2)$ where the states s_1, s_2, s_3, s_4 are the cart position, cart velocity, pendulum velocity and pendulum angular position, and it will be maximum if the pendulum is swing up to π angle with zero velocity. We evaluate the linear policy $\pi(s) = As+b$ where $A \in \mathbb{R}^{1\times 4}$ and $b \in \mathbb{R}^{1\times 1}$. The batch size is set to be 20. The stepsize parameters are chosen in range $\{\eta, n_0\} \in \{0.05, 0.2, 1, 10, 50, 100\}$. We adopted Gaussian kernel and the primal and dual kernel bandwidth are se lected by median trick with coeffcient in $\{0.5, 1, 5, 10\}$. The γ is set to be 0.96. Results are reported in Figure 5.3(b).

PUMA-560 manipulation PUMA-560 is a robotic arm that has 6 degrees of freedom with 6 actuators on each joint. The task is to steer the end-effector to the desired position and orientation with zero velocity. The reward is $R(s) = \frac{1}{2}(\sum_{i=1}^{4}(s_i - \frac{\pi}{4})^2 + \sum_{i=5}^{6}(s_i + \frac{\pi}{4})^2 + \sum_{i=7}^{12} s_i^2)$ where $s_1, \ldots, s_6$ and $s_7, \ldots, s_{12}$ are joint angles and velocities, respectively, and it will be maximum if the arm is located to the desired position. We evaluate the linear policy $\pi(s) = As + b$ where $A \in \mathbb{R}^{6\times 12}$ and $b \in \mathbb{R}^{6\times 1}$. The batch size is set to be 20. The stepsize parameters are chosen in range $\{\eta, n_0\} \in \{0.05, 0.1, 5, 10, 100, 500\}$. We adopted Gaussian kernel and the primal and dual kernel bandwidth are selected by median trick with coeffcient in $\{0.5, 1, 5, 10\}$. The γ is set to be 0.9. Results are reported in Figure 5.3(c).

In all experiments, the proposed algorithm performs consistently better than the competitors. The advantages of proposed algorithm mainly come from three aspects: **i)**, it utilizes more flexible dual function space, rather than the constrained space in GTD2; **ii)**, it directly optimizes the MSBE, rather than its surrogate as in GTD2 and RG; **iii)**, it directly targets on value function estimation and forms an one-shot algorithm, rather than a two-stage procedure in kernel MDP including estimating conditional kernel embedding as an intermediate step.

5.7 Summary

The optimization view with the conditional integral operator provides us a unified framework for *learning over dynamics* problem, which includes learning with invaraince, policy evaluation, events prediction in stochastic processes, and so on.

We propose a novel sample-efficient algorithm, *Embedding-SGD*, which benefits from a fresh employ of saddle point by *dual embedding* technique, to mitigate the difficulty with limited samples from conditional distribution as well as the presence of nested expectations. To our best knowledge, among all existing algorithms able to solve such problems, this is *the first* algorithm that allows to take only one sample at a time from the conditional distribution and comes with provable theoretical guarantee. The proposed algorithm achieves the state-of-the-art performances on the tasks of learning with invariance and policy evaluation in reinforcement learning comparing to the existing algorithms. As we discussed in Appendix C.6, the algorithm is also applicable to actor-critic framework for policy optimization.

In addition to its wide applicability, our algorithm is also very versatile and amenable for all kinds of enhancement. The algorithm can be easily extended with random feature approximation or doubly stochastic gradient trick. Moreover, we can extend the framework by learning neural network embedding in Appendix C.5.3. It should be emphasized that since the primal and dual function spaces are designed for different purposes, although we use both RKHS in main text for simplicity, we can also use different function approximators separately for primal and dual functions.

CHAPTER 6
DUAL EMBEDDING FOR SMOOTHED BELLMAN ERROR

In reinforcement learning (RL), the goal of an agent is to learn a policy that maximizes long-term returns by sequentially interacting with an unknown environment [2]. Within the Markov decision processes assumption, the optimal policy can be obtained by matching the Bellman optimality equation. When function approximation is used, solving the Bellman optimality equation with stability guarantees has remained a major open problem in reinforcement learning for decades. The fundamental difficulty is that the Bellman operator may become an expansion in general, resulting in oscillating and even divergent behavior of popular algorithms like Q-learning.

Different from the policy evaluation problem, due to the max-operator in the Bellman optimality equation, the mean-square error for Bellman optimality equation is *not* a special case of the optimization with the conditional integral operator. Directly applying the dual embedding technique will lead to unstable performance due to the max-operator in practice.

We revisit the Bellman optimality equation by Nesterov's smoothing technique. By exploiting the property in the smoothed Bellman equation, we obtain a surrogate loss which can be reduced to the optimization with the conditional integral operator we discussed in Chapter 5. We then develop a new algorithm, called *Smoothed Bellman Error Embedding*, to solve this optimization problem where any differentiable function class may be used. We provide what we believe to be the first convergence guarantee for general nonlinear function approximation, and rigoriously analyze the sample complexity and extra induced error of the algorithm. Empirically, our algorithm compares favorably to state-of-the-art baselines in several benchmark control problems.

6.1 Introduction

In the Markov decision process, the optimal value function are characterized as a fixed point of the Bellman operator. A fundamental result for MDP is that the Bellman operator is a

contraction in the value-function space, so the optimal value function is the unique fixed point. Furthermore, starting from any initial value function, iterative applications of the Bellman operator ensure convergence to the fixed point. Interested readers are referred to the textbook of [156] for details.

Many of the most effective RL algorithms have their root in such a fixed-point view. The most prominent family of algorithms is perhaps the temporal-difference algorithms, including TD(λ) [157], Q-learning [158], SARSA [159, 160], and numerous variants such as the empirically very successful DQN [161] and A3C [162] implementations. Compared to direct policy search/gradient algorithms like REINFORCE [163], these fixed-point methods make learning more efficient by *bootstrapping* (a sample-based version of Bellman operator).

When the Bellman operator can be computed exactly (even on average), such as when the MDP has finite state/actions, convergence is guaranteed thanks to the contraction property [164]. Unfortunately, when function approximatiors are used, such fixed-point methods *easily* become unstable or even divergent [165, 28, 166], except in a few special cases. For example,

- for some rather restrictive function classes, such as those with a non-expansion property, some of the finite-state MDP theory continues to apply with modifications [167, 168, 169];
- when *linear* value function approximation in certain cases, convergence is guaranteed: for evaluating a *fixed* policy from *on-policy* samples [166], for evaluating the policy using a closed-form solution from *off-policy* samples [170, 171], or for optimizing a policy using samples collected by a stationary policy [172].

In recent years, a few authors have made important progress toward finding scalable, convergent TD algorithms, by designing proper objective functions and using stochastic gradient descent (SGD) to optimize them [139, 173]. Later on, it was realized that several of these gradient-based algorithms can be interpreted as solving a primal-dual problem [174, 140, 175, 13]. This insight has led to novel, faster, and more robust algorithms by adopting sophisticated optimization techniques [176]. Unfortunately, to the best of our knowledge, all existing works either assume linear function approximation or are designed for policy evalu-

ation. It remains a major open problem how to find the *optimal policy* reliably with general *nonlinear* function approximators such as neural networks, especially in the presence of *off-policy* data.

6.1.1 Contributions

In this work, we take a substantial step towards solving this decades-long open problem, leveraging the proposed framework with conditional integral operator, to derive a new algorithm called *Smoothed Bellman Error Embedding (SBEED) algorithm*. Our development hinges upon a novel view of a smoothed Bellman optimality equation, which is a special case of the optimization (5.1), then transformed to the final primal-dual optimization problem. SBEED learns the optimal value function and a stochstic policy in the primal, and the Bellman error (also known as Bellman residual) in the dual. By doing so, it avoids the non-smooth max-operator in the Bellman operator, as well as the double-sample challenge that has plagued RL algorithm designs [28]. More specifically,

- SBEED is stable for a broad class of nonlinear function approximators including neural networks, and provably converges to a solution with vanishing gradient. This holds even in the more challenging off-policy case;
- it uses bootstrapping to yield high sample efficiency, as in TD-style methods, and is also generalized to cases of multi-step bootstrapping and eligibility traces;
- it avoids the double-sample issue and directly optimizes the squared Bellman error based on sample trajectories;
- it uses stochastic gradient descent to optimize the objective, thus very efficient and scalable.

Furthermore, the algorithm handles both the optimal value function estimation and policy optimization in a unified way, and readily applies to both continuous and discrete action spaces. We compare the algorithm with state-of-the-art baselines on several continuous control benchmarks, and obtain excellent results.

6.2 Preliminary

In this section, we introduce notation and technical background that is needed. Since we will apply the dual embedding technique to the particular policy optimization problem in Markov decision process (MDP), we will provide a brief introduction to the MDP. To make the thesis consistent with the common notations in MDP community, we denote a MDP as $\mathbb{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is a (possible infinite) state space, $\mathcal{A}$ is an action space, $P(\cdot|s,a)$ the transition probability defining the distribution over next states upon taking action a on state s, $R(s,a)$ the average immediate reward by taking action a in state s, and $\gamma \in (0,1)$ a discount factor. Given an MDP, we wish to find a possibly stochastic policy $\pi : \mathcal{S} \to \mathcal{P}_{\mathcal{A}}$ to maximize the expected discounted cumulative reward starting from any state $s \in \mathcal{S}$: $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s, \pi\right]$, where $\mathcal{P}_{\mathcal{A}}$ denotes all probability measures over $\mathcal{A}$. The set of all policies is denoted by $\mathcal{P} := (\mathcal{P}_{\mathcal{A}})^{\mathcal{S}}$.

Define $V^*(s) := \max_{\pi(\cdot|s)} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, \pi\right]$ to be the optimal value function. It is known that V^* is the unique fixed point of the Bellman operator $\mathcal{T}_B$, or equivalently, the unique solution to the Bellman optimality equation (Bellman equation, for short) [156]:

$$V(s) = (\mathcal{T}_B V)(s) := \max_a \; R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right]. \tag{6.1}$$

The optimal policy π^* is related to V^* by the following:

$$\pi^*(a|s) = \operatorname*{argmax}_a \left\{ R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V^*(s')\right] \right\}.$$

It should be noted that in practice, for convenience we often work on the Q-function instead of the state-value function V^*. In this section, it suffices to use the simpler V^* function.

6.3 A Primal-Dual View of Bellman Equation

In this section, we introduce a novel view of Bellman equation that enables the development of the new algorithm in Section 6.4. After reviewing the Bellman equation and the challenges to solve it, we describe the two key technical ingredients that lead to our primal-dual

reformulation.

We start with another version of Bellman equation that is equivalent to Eq (6.1) (see, e.g., [156]):

$$V(s) = \max_{\pi(\cdot|s)\in\mathcal{P}_{\mathcal{A}}} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V(s') \right] \right]. \tag{6.2}$$

Eq (6.2) makes the role of a policy explicit. Naturally, one may try to jointly optimize over V and π to minimize the discrepancy between the two sides of (6.2). For concreteness, we focus on the square distance in this dissertation, but our results can be extended to other convex loss functions. Let μ be some given state distribution so that $\mu(s) > 0$ for all $s \in \mathcal{S}$. Minimizing the *squared Bellman error* gives the following:

$$\min_{V} \mathbb{E}_{s\sim\mu} \left[\left(\max_{\pi(\cdot|s)\in\mathcal{P}_{\mathcal{A}}} \mathbb{E}_{a\sim\pi(\cdot|s)} \left[R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V(s') \right] \right] - V(s) \right)^2 \right]. \tag{6.3}$$

While natural, this approach has several major difficulties when it comes to optimization, which are to be dealt with in the following subsections:

- The max operator over $\mathcal{P}_{\mathcal{A}}$ introduces non-smoothness to the objective function. A slight change in V may cause large differences in the RHS of Eq (6.2).
- The conditional expectation, $\mathbb{E}_{s'|s,a}[\cdot]$, composed within the square loss, requires double samples [28] to obtain unbiased gradients, which is often impractical in most but simulated environments.

6.3.1 Smoothed Bellman Equation

To avoid the instability and discontinuity caused by the max operator, we use the smoothing technique of [177] to smooth the Bellman operator $\mathcal{T}_B$. Since policies are conditional distributions over $\mathcal{A}$, we choose entropy regularization, and Eq (6.2) becomes:

$$V_\lambda(s) = \max_{\pi(\cdot|s)\in\mathcal{P}_{\mathcal{A}}} \left(\mathbb{E}_{a\sim\pi(\cdot|s)} \big(R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V_\lambda(s') \right] \big) + \lambda H(\pi, s) \right), \tag{6.4}$$

where $H(\pi, s) := -\sum_{a\in\mathcal{A}} \pi(a|s) \log \pi(a|s)$, and $\lambda \geqslant 0$ controls the degree of smoothing. Note that with $\lambda = 0$, we obtain the standard Bellman equation. Moreover, the regulariza-

tion may be viewed a shaping reward added to the reward function of an induced, equivalent MDP; see Appendix D.3.2 for more details.

Since negative entropy is the conjugate of the log-sum-exp function [178, Example 3.25], Eq (6.4) can be written equivalently as

$$V_\lambda(s) = (\mathcal{T}_\lambda V_\lambda)(s) := \lambda \log \left(\sum_{a \in \mathcal{A}} \exp \left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a} [V_\lambda(s')]}{\lambda} \right) \right) \tag{6.5}$$

where the log-sum-exp is an effective smoothing approximation of the max-operator.

Remark: While Eqns (6.4) and (6.5) are inspired by Nestorov smoothing technique, they can also be derived from other principles [179, 180, 181, 182, 183]. For example, [182] uses entropy regularization in the policy space to encourage exploration, but arrives at the same smoothed form; the smoothed operator $\mathcal{T}_\lambda$ is called "Mellowmax" by [183], which is obtained as a particular instantiation of the quasi-arithmetic mean. In the rest of the subsection, we review the properties of $\mathcal{T}_\lambda$, although some of the results have appeared in the literature in slightly different forms. Proofs are deferred to Appendix D.1.

First, we show $\mathcal{T}_\lambda$ is also a contraction, as with the standard Bellman operator [180, 183]:

Proposition 24 (Contraction) *$\mathcal{T}_\lambda$ is a γ-contraction. Consequently, the corresponding smoothed Bellman equation (6.4), or equivalently (6.5), has a unique solution V^*_λ.*

Second, we show that while in general $V^* \neq V^*_\lambda$, their difference is controlled by λ. To do so, define $H^* := \max_{s \in \mathcal{S}, \pi(\cdot|s) \in \mathcal{P}_\mathcal{A}} H(\pi, s)$. For finite action spaces, we immediately have $H^* = \log(|\mathcal{A}|)$.

Proposition 25 (Smoothing bias) *Let V^* and V^*_λ be fixed points of (6.2) and (6.4), respectively. Then,*

$$\|V^*(s) - V^*_\lambda(s)\|_\infty \leqslant \frac{\lambda H^*}{1-\gamma}.$$

Consequently, as $\lambda \to 0$, V^*_λ converges to V^* pointwisely.

Finally, the smoothed Bellman operator has the very nice property of temporal consistency [179, 182]:

Proposition 26 (Temporal consistency) *Assume $\lambda > 0$. Let V_λ^* be the fixed point of (6.4) and π_λ^* the corresponding policy that attains the maximum on the RHS of (6.4). Then, $(V_\lambda^*, \pi_\lambda^*)$ is the unique (V, π) pair that satisfies the following equality for all $(s, a) \in \mathcal{S} \times \mathcal{A}$:*

$$V(s) = R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right] - \lambda \log \pi(a|s)\,. \tag{6.6}$$

In other words, Eq (6.6) provides an easy-to-check condition to characterize the optimal value function and optimal policy on *arbitrary* pair of (s, a), therefore, which is easy to incorporate *off-policy* data. It can also be extended to the multi-step or eligibility-traces cases (Appendix D.3; see also [2, Chapter 7]). Later, this condition will be one of the critical foundations to develop our new algorithm.

6.3.2 Bellman Error Embedding

A natural objective function inspired by (6.6) is the *mean squared consistency Bellman error*, given by:

$$\min_{V,\pi\in\mathcal{P}} \ell(V,\pi) \quad := \quad \mathbb{E}_{s,a}\Big[\big(R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V(s')\right] - \lambda \log \pi(a|s) - V(s)\big)^2\Big]\,, \tag{6.7}$$

where $\mathbb{E}_{s,a}[\cdot]$ is shorthand for $\mathbb{E}_{s\sim\mu(\cdot),a\sim\pi_b(\cdot|s)}[\cdot]$. With finite $\mathcal{S}$ and $\mathcal{A}$, the optimization (6.7) provides exact solution to the temporal condition (6.6), while with infinite $\mathcal{S}$ or $\mathcal{A}$, such an objective converts the point-wise equivalence in the temporal consistency (6.6) to the equivalence in L_2-space implicitly. Due to the inner conditional expectation, it would require two independent sample of s' (starting from the same (s, a)) to obtain an unbiased estimate of gradient of f, a problem known as the double-sample issue [28], the same as we discussed in Chapter 5. In practice, however, one can rarely obtain two independent samples except in simulated environments.

Recall the optimization (6.7) is a speical case of the conditional integral operator as we discussed in Chapter 5, we can use the *dual embedding* techinque, *i.e.*, using of the conjugate of the square function [178]: $x^2 = \max_\nu \left(2\nu x - \nu^2\right)$, as well as the interchangeability

principle in Lemma 20 to rewrite the optimization problem (6.7) into an equivalent form:

$$\min_{V,\pi\in\mathcal{P}} \max_{\nu\in\mathcal{F}(\mathcal{S}\times\mathcal{A})} L(V,\pi;\nu) \;:=\; 2\mathbb{E}_{s,a,s'}\Big[\nu(s,a)\big(R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)-V(s)\big)\Big] \\ -\mathbb{E}_{s,a,s'}\left[\nu^2(s,a)\right], \tag{6.8}$$

where $\mathcal{F}(\mathcal{S}\times\mathcal{A})$ is the set of real-valued functions on $\mathcal{S}\times\mathcal{A}$ and $\mathbb{E}_{s,a,s'}[\cdot]$ is shorthand for $\mathbb{E}_{s\sim\mu(\cdot),a\sim\pi_b(\cdot|s),s'\sim P(\cdot|s,a)}[\cdot]$. However, different from the optimizations in Chapter 5, (6.8) is not a standard convex-concave saddle-point problem: the objective is convex in V for any fixed (π,ν), and concave in ν for any fixed (V,π), but not necessarily convex in $\pi\in\mathcal{P}$ for any fixed (V,ν). Therefore, although the Embedding-SGD is applicable, the convergence property may not hold. We design new algorithm, which still preserve the convergence, in Section 6.4.

Remark: In contrast to our min-max formulation (6.8), [182] get around the double-sample obstacle by minimizing an upper bound of $\ell(V,\pi)$:

$$\tilde{\ell}(V,\pi) := \mathbb{E}_{s,a,s'}\left[\big(R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)-V(s)\big)^2\right].$$

As is known [28], the gradient of $\tilde{\ell}$ is different from that of f, as it has a conditional variance term coming from the stochastic outcome s'. In problems where this variance is highly heterogeneous across different (s,a) pairs, impact of such a bias can be substantial.

Finally, reparametrizing the dual function $\nu(s,a)=\varsigma(s,a)-V(s)$ and substituting into the objective in the min-max problem, we obtain

$$\min_{V,\pi}\max_{\varsigma\in\mathcal{F}(\mathcal{S}\times\mathcal{A})} L_1(V,\pi;\varsigma) := \mathbb{E}_{s,a,s'}\left[\big(\delta(s,a,s')-V(s)\big)^2\right]-\mathbb{E}_{s,a,s'}\left[\big(\delta(s,a,s')-\varsigma(s,a)\big)^2\right], \tag{6.9}$$

where $\delta(s,a,s') := R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)$. Note that the first term is $\tilde{\ell}(V,\pi)$, and the second term will cancel the extra variance term (see Proposition 55 in Appendix D.2). The use of an auxiliary function to cancel the variance is also observed by [169]. On the other hand, when function approximation is used, extra bias will also be introduced. We note that such a min-max view of debiasing the extra variance term leads to a useful mechanism

for better bias-variance trade-offs, leading to the final primal-dual formulation we aim to solve in the next section:

$$\min_{V,\pi\in\mathcal{P}_\varsigma}\max_{\varsigma\in\mathcal{F}(\mathcal{S}\times\mathcal{A})} L_\eta(V,\pi;\varsigma)=\mathbb{E}_{s,a,s'}\left[\left(\delta(s,a,s')-V(s)\right)^2\right]-\eta\mathbb{E}_{s,a,s'}\left[\left(\delta(s,a,s')-\varsigma(s,a)\right)^2\right], \tag{6.10}$$

where $\eta\in[0,1]$ is a hyper-parameter controlling the trade-off. When $\eta=1$, this reduces to the original min-max formulation (6.8). When $\eta=0$, this reduces to the surrogate objective considered by [182].

6.4 Smoothed Bellman Error Embedding

In this section, we derive the *Smoothed Bellman Error EmbeDding* (SBEED) algorithm, based on stochastic mirror descent [70], to solve the smoothed Bellman equation. For simplicity of exposition, we mainly discuss the one-step optimization (6.10), although it is possible to generalize the algorithm to the multi-step and eligibility-traces settings; see Appendices D.3.2 and D.3.3 for details.

Due to the curse of dimensionality, the quantities (V,π,ς) are often represented by compact, parametric functions in practice. Denote these parameters by $w=(w_V,w_\pi,w_\varsigma)$. Abusing notation a little bit, we now write the objective function $L_\eta(V,\pi;\varsigma)$ as $L_\eta(w_V,w_\pi;w_\varsigma)$.

First, we note that the inner (dual) problem is standard least-squares regression with parameter w_ς, so can be solved using a variety of algorithms [184]; in the presence of special structures like convexity, global optima can be found efficiently [178]. The more involved part is to optimize the primal (w_V,w_π), whose gradients are given by the following theorem.

Theorem 27 (Primal gradient) *Denote $\bar{\ell}_\eta(w_V,w_\pi):=L_\eta(w_V,w_\pi;w_\varsigma^*)$ as well as $w_\varsigma^*=\arg\max_{w_\varsigma}L_\eta(w_V,w_\pi;w_\varsigma)$, let $\delta_{s,a,s'}$ be a shorthand for $\delta(s,a,s')$, and $\hat{\varsigma}$ be dual parameterized by w_ς^*. Then,*

$$\begin{aligned}\nabla_{w_V}\bar{\ell}_\eta =&2\mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'}-V(s)\right)\left(\gamma\nabla_{w_V}V(s')-\nabla_{w_V}V(s)\right)\right]\\ &-2\eta\gamma\mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'}-\hat{\varsigma}(s,a)\right)\nabla_{w_V}V(s')\right],\\ \nabla_{w_\pi}\bar{\ell}_\eta =&-2\lambda\mathbb{E}_{s,a,s'}\left[\left((1-\eta)\delta_{s,a,s'}+\eta\hat{\varsigma}(s,a)-V(s)\right)\cdot\nabla_{w_\pi}\log\pi(a|s)\right].\end{aligned}$$

With gradients given above, we may apply stochastic mirror descent to update w_V and w_π; that is, given a stochastic gradient direction (for either w_V or w_π), we solve the following prox-mapping in each iteration,

$$\begin{aligned} P_{z_V}(g) &= \operatorname*{argmin}_{w_V} \langle w_V, g\rangle + D_V(w_V, z_V), \\ P_{z_\pi}(g) &= \operatorname*{argmin}_{w_\pi} \langle w_\pi, g\rangle + D_\pi(w_\pi, z_\pi). \end{aligned}$$

where z_V and z_π can be viewed the current weight, and $D_V(w, z)$ and $D_\pi(w, z)$ are Bregman divergences. We can use Euclidean metric for both w_V and w_π, and possibly KL-divergence for w_π. The per-iteration computation complexity is therefore very low, and the algorithm can be scaled up to complex nonlinear approximations.

Algorithm 5 illustrates SBEED, combined with experience replay [185] for greater data efficiency, in an online RL setting. New samples are added to the experience replay buffer $\mathcal{D}$ at the beginning of each episode (Lines 3–5) with a behavior policy. Lines 6–11 correspond to the stochastic mirror descent updates on the primal parameters. Line 12 sets the behavior policy to be the current policy estimate, although other choices may be used. For example, π_b can be a fixed policy [169], which is the case we will analyze in the next section.

Remark (Role of dual variables): The dual variable is obtained by solving

$$\min_{\varsigma} \mathbb{E}_{s,a,s'} \left[\left(R(s,a) + \gamma V(s') - \lambda \log \pi(a|s) - \varsigma(s,a) \right)^2 \right].$$

The solution to this optimization problem is

$$\varsigma^*(s,a) = R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V(s') \right] - \lambda \log \pi(a|s).$$

Therefore, the dual variables try to approximate the one-step smoothed Bellman backup values, given a (V, π) pair. Similarly, in the equivalent (6.8), the optimal dual variable $\nu(s,a)$ is to fit the one-step smoothed Bellman error. Therefore, each iteration of SBEED could be understood as first fitting a parametric model to the one-step Bellman backups (or equivalently, the one-step Bellman error), and then applying stochastic mirror descent

Algorithm 5 Online SBEED learning with experience replay

Initialize $w = (w_V, w_\pi, w_\varsigma)$ and π_b randomly, set ϵ.
for episode $i = 1, \ldots, T$ **do**
  for size $k = 1, \ldots, K$ **do**
    Add new transition (s, a, r, s') into $\mathcal{D}$ by executing behavior policy π_b.
  end for
  for iteration $j = 1, \ldots, N$ **do**
    Update w_ς^j by solving

$$\min_{w_\varsigma} \widehat{\mathbb{E}}_{\{s,a,s'\}\sim\mathcal{D}} \left[\big(\delta(s,a,s') - \varsigma(s,a)\big)^2 \right].$$

    Decay the stepsize ζ_j in rate $\mathcal{O}(1/j)$.
    Compute the stochastic gradients w.r.t. w_V and w_π as $\widehat{\nabla}_{w_V} \bar{\ell}(V, \pi)$ and $\widehat{\nabla}_{w_\pi} \bar{\ell}(V, \pi)$.
    Update the parameters of primal function by solving the prox-mappings, *i.e.*,

$$\text{update } V: \quad w_V^j = P_{w_V^{j-1}}(\zeta_j \widehat{\nabla}_{w_V} \bar{\ell}(V, \pi))$$
$$\text{update } \pi: \quad w_\pi^j = P_{w_\pi^{j-1}}(\zeta_j \widehat{\nabla}_{w_\pi} \bar{\ell}(V, \pi))$$

  end for
  Update behavior policy $\pi_b = \pi^N$.
end for

to adjust V and π.

Remark (Connection to TRPO and NPG): The update of w_π is related to trust region policy optimization (TRPO) [186] and natural policy gradient (NPG) [187, 188] when D_π is the KL-divergence. Specifically, in [187] and [188], w_π is updated by setting as $\operatorname{argmin}_{w_\pi} \mathbb{E}\left[\left\langle w_\pi, \nabla_{w_\pi} \log \pi^t(a|s) A(a,s) \right\rangle\right] + \frac{1}{\eta} \mathrm{KL}(\pi_{w_\pi} || \pi_{w_\pi^{old}})$, which is similar to $P_{w_\pi^{j-1}}$ with the difference in replacing the $\log \pi^t(a|s) A(a,s)$ with our gradient. In [186], a related optimization with hard constraints is used for policy updates: $\min_{w_\pi} \mathbb{E}\left[\pi(a|s) A(a,s)\right]$, such that $\mathrm{KL}(\pi_{w_\pi} || \pi_{w_\pi^{old}}) \leqslant \eta$. Although these operations are similar to $P_{w_\pi^{j-1}}$, we emphasize that the estimation of the advantage function, $A(s,a)$, and the update of policy are separated in NPG and TRPO. Arbitrary policy evaluation algorithm can be adopted for estimating the value function for *current* policy. While in our algorithm, $(1-\eta)\delta(s,a) + \eta\varsigma^*(s,a) - V(s)$ is different from the vanilla advantage function, which is designed appropriate for off-policy particularly, and the estimation of $\varsigma(s,a)$ and $V(s)$ is also integrated as the whole part.

6.5 Theoretical Analysis

In this section, we give a theoretical analysis for our algorithm in the same setting of [169] where samples are prefixed and from *one single β-mixing off-policy sample path*. For simplicity, we consider the case that applying the algorithm for $\eta = 1$ with the equivalent optimization (6.8). The analysis is applicable to (6.9) directly. There are three groups of results. First, in Section 6.5.1, we show that under appropriate choices of stepsize and prox-mapping, SBEED converges to a stationary point of the finite-sample approximation (i.e., empirical risk) of the optimization (6.8). Second, in Section 6.5.2, we analyze generalization error of SBEED. Finally, in Section 6.5.3, we give an overall performance bound for the algorithm, by combining four sources of errors: (i) optimization error, (ii) generalization error, (iii) bias induced by Nesterov smoothing, and (iv) approximation error induced by using function approximation.

Table 6.1: The notations for different objectives in the analysis of SBEED algorithm

	minimax obj.	primal obj.	optimum
original	$L(V, \pi; \nu)$	$\ell(V, \pi)$	$(V^*_\lambda, \pi^*_\lambda)$
parametric	$L_w(V_w, \pi_w; \nu_w)$	$\ell_w(V_w, \pi_w)$	(V^*_w, π^*_w)
empirical	$\widehat{L}_T(V_w, \pi_w; \nu_w)$	$\widehat{\ell}_T(V_w, \pi_w)$	$(\widehat{V}^*_w, \widehat{\pi}^*_w)$

Notations Denote by $\mathcal{V}_w$, $\mathcal{P}_w$ and $\mathcal{H}_w$ the parametric function classes of value function V, policy π, and dual variable ν, respectively. Denote the total number of steps in the given off-policy trajectory as T. We summarize the notations for the objectives after parametrization and finite-sample approximation and their corresponding optimal solutions in Table 6.1 for reference.

Denote the L_2 norm of a function f w.r.t. $\mu(s)\pi_b(a|s)$ by $\|f\|^2 := \int f(s,a)^2 \mu(s)\pi_b(a|s)dsda$. We introduce a scaled norm : $\|V\|^2_{\mu\pi_b} = \int \left(\gamma \mathbb{E}_{s'|s,a}\left[V(s')\right] - V(s)\right)^2 \mu(s)\pi_b(a|s)dsda$; for value function; this is indeed a well-defined norm since $\|V\|^2_{\mu\pi_b} = \|(\gamma P - I)V\|_2^2$ and $I - \gamma P$ is injective.

6.5.1 Convergence Analysis

It is well-known that for convex-concave saddle point problems, applying stochastic mirror descent ensures global convergence in a sublinear rate [70]. However, this no longer holds for problems without convex-concavity. Our SBEED algorithm, on the other hand, can be regarded as a special case of the stochastic mirror descent algorithm for solving the non-convex primal minimization problem $\min_{V_w,\pi_w} \widehat{\ell}_T(V_w,\pi_w)$. The latter was proven to converge sublinearly to a stationary point when stepsize is diminishing and Euclidean distance is used for the prox-mapping [189]. For completeness, we list the result below.

Theorem 28 (Convergence, [189]) *Consider the case when Euclidean distance is used in the algorithm. Assume that the parametrized objective $\widehat{\ell}_T(V_w,\pi_w)$ is K-Lipschitz and variance of its stochastic gradient is bounded by σ^2. Let the algorithm run for N iterations with stepsize $\zeta_k = \min\{\frac{1}{K}, \frac{D'}{\sigma\sqrt{N}}\}$ for some $D' > 0$ and output $w^1,\dots,w^N$. Setting the candidate solution to be $(\widehat{V}_w^N, \widehat{\pi}_w^N)$ with w randomly chosen from $w^1,\dots,w^N$ such that $P(w=w^j) = \frac{2\zeta_j - K\zeta_j^2}{\sum_{j=1}^N (2\zeta_j - K\zeta_j^2)}$, then it holds that $\mathbb{E}\left[\left\|\nabla \widehat{\ell}_T(\widehat{V}_w^N, \widehat{\pi}_w^N)\right\|^2\right] \leqslant \frac{KD^2}{N} + (D' + \frac{D}{D'})\frac{\sigma}{\sqrt{N}}$ where $D := \sqrt{2(\widehat{\ell}_T(V_w^1,\pi_w^1) - \min \widehat{\ell}_T(V_w,\pi_w))/K}$ represents the distance of the initial solution to the optimal solution.*

The above result implies that the algorithm converges sublinearly to a stationary point, whose rate will depend on the smoothing parameter.

In practice, once we parametrize the dual function, ν or ς, with neural networks, we cannot achieve the optimal parameters. However, we can still achieve convergence by applying the stochastic gradient descent to a (statistical) local Nash equilibrium asymptotically under mild conditions. We provided the detailed Algorithm 12 and the convergence analysis in Appendix D.4.3.

6.5.2 Statistical Error

In this section, we characterize the statistical error, namely, $\epsilon_{\text{stat}}(T) := \ell_w(\widehat{V}_w^*, \widehat{\pi}_w^*) - \ell_w(V_w^*, \pi_w^*)$, induced by learning with finite samples. We first make the following standard assumptions about the MDPs:

Assumption 10 (MDP regularity) *Assume* $\|R(s,a)\|_\infty \leqslant C_R$ *and that there exists an optimal policy,* $\pi^*_\lambda(a|s)$*, such that* $\|\log \pi^*_\lambda(a|s)\|_\infty \leqslant C_\pi$*.*

Assumption 11 (Sample path property, [169]) *Denote* $\mu(s)$ *as the stationary distribution of behavior policy* π_b *over the MDP. We assume* $\pi_b(a|s) > 0$*,* $\forall\,(s,a) \in \mathcal{S} \times \mathcal{A}$*, and the corresponding Markov process* $P^{\pi_b}(s'|s)$ *is ergodic. We further assume that* $\{s_i\}_{i=1}^T$ *is strictly stationary and exponentially* β*-mixing with a rate defined by the parameters* (b,κ)[1]*.*

Assumption 10 ensures the solvability of the MDP and boundedness of the optimal value functions, V^* and V^*_λ. Assumption 11 ensures β-mixing property of the samples $\{(s_i,a_i,R_i)\}_{i=1}^T$ (see e.g., Proposition 4 in [190]) and is often necessary to prove large deviation bounds.

Invoking a generalized version of Pollard's tail inequality to β-mixing sequences and prior results in [169] and [191], we show that

Theorem 29 (Statistical error) *Under Assumption 2, it holds with at least probability* $1-\delta$*,*

$$\epsilon_{stat}(T) \leqslant 2\sqrt{\frac{M\left(\max\left(M/b,1\right)\right)^{1/\kappa}}{C_2 T}},$$

where M, C_2 *are some constants.*

Detailed proof can be found in Appendix D.4.2.

6.5.3 Error Decomposition

As one shall see, the error between $(\widehat{V}_w^N, \widehat{w}^N)$ (optimal solution to the finite sample problem) and the true optimal (V^*, π^*) to the Bellman equation consists three parts: i) the error introduced by smoothing, which has been characterized in Section 6.3.1, ii) the approximation error, which is tied to the flexibility of the parametrized function classes $\mathcal{V}_w$, $\mathcal{P}_w$, $\mathcal{H}_w$, and iii) the statistical error. More specifically, we arrive at the following explicit decomposition:

Denote $\epsilon^\pi_{\text{app}} := \sup_{\pi\in\mathcal{P}} \inf_{\pi'\in\mathcal{P}_w} \|\pi - \pi'\|_\infty$ as the function approximation error between $\mathcal{P}_w$ and $\mathcal{P}$. Denote ϵ^V_{app} and $\epsilon^\nu_{\text{app}}$ as approximation errors for V and ν, accordingly. More specifically, we arrive at

[1] A β-mixing process is said to mix at an exponential rate with parameter $b, \kappa > 0$ if $\beta_m = O(\exp(-bm^{-\kappa}))$.

Theorem 30 *Under Assumptions 1 and 2, it holds that* $\left\|\widehat{V}_w^N - V^*\right\|_{\mu\pi_b}^2 \leqslant 12(K+C_\infty)\epsilon_{app}^\nu + 2C_\nu(1+\gamma)\epsilon_{app}^V(\lambda)+6C_\nu\epsilon_{app}^\pi(\lambda)+16\lambda^2C_\pi^2+\left(2\gamma^2+2\right)\left(\frac{\gamma\lambda}{1-\gamma}H^*\right)^2+2\epsilon_{stat}(T)+2\left\|\widehat{V}_w^N-\widehat{V}_w^*\right\|_{\mu\pi_b}^2$, *where* $C_\infty = \max\left\{\frac{C_R}{1-\gamma}, C_\pi\right\}$ *and* $C_\nu = \max_{\nu\in\mathcal{H}_w}\|\nu\|_2$.

Detailed proof can be found in Appendix D.4.1. Ignoring the constant factors, the above results can be simplified as

$$\left\|\widehat{V}_w^N - V^*\right\|_{\mu\pi_b}^2 \leqslant \epsilon_{\text{app}}(\lambda) + \epsilon_{\text{sm}}(\lambda) + \epsilon_{\text{stat}}(T) + \epsilon_{\text{opt}},$$

where $\epsilon_{\text{app}}(\lambda) := \mathcal{O}(\epsilon_{\text{app}}^\nu + \epsilon_{\text{app}}^V(\lambda) + \epsilon_{\text{app}}^\pi(\lambda))$ corresponds to the approximation error, $\epsilon_{\text{sm}}(\lambda) := \mathcal{O}(\lambda^2)$ corresponds to the bias induced by smoothing, and $\epsilon_{\text{stat}}(T) := \mathcal{O}(1/\sqrt{T})$ corresponds to the statistical error.

There exists a delicate trade-off between the smoothing bias and approximation error. Using large λ increases the smoothing bias but decreases the approximation error since the solution function space is better behaved. The concrete correspondence between λ and $\epsilon_{\text{app}}(\lambda)$ depends on the specific form of the function approximators, which is beyond the scope of this thesis. Finally, when the approximation is good enough (i.e., zero approximation error and full column rank of feature matrices), then our algorithm will converge to the optimal value function V^* as $\lambda \to 0$, $N, T \to \infty$.

6.6 Connections to Other Approaches

One of our main contributions is a provably convergent algorithm when nonlinear approximation is used in the off-policy control case. Convergence guarantees exist in the literature for a few rather special cases, as reviewed in the introduction [165, 167, 166, 168, 169, 192]. Of particular interest is the Greedy-GQ algorithm [172], who uses two time-scale analysis to shown asymptotic convergence only for *linear* function approximation in the controlled case. However, it does not take the true gradient estimator in the algorithm, and the update of policy may become intractable when the action space is continuous.

Algorithmically, our method is most related to RL algorithms with entropy-regularized policies. Different from the motivation in our method where the entropy regularization is in-

troduced in dual form for smoothing [177], the entropy-regularized MDP has been proposed for exploration [193, 194], taming noise in observations [195, 180], and ensuring tractability [132]. Specifically, [180] proposed soft Q-learning for the tabular case, but its extension to the function approximation case is hard, as the summation operation in log-sum-exp of the update rule becomes a computationally expensive integration. To avoid such a difficulty, [194] approximate the integral via Monte-Carlo method with Stein variational gradient descent sampler, but limited theory is provided. Another related algorithm is developed by [183] for the tabular case, which resembles SARSA with a particular policy; also see [196] for a Bayesian variant. Observing the duality connection between soft Q-learning and maximum entropy policy optimization, [181] and [197] investigate the equivalence between these two types of algorithms.

Besides the difficulty to generalize these algorithms to multi-step trajectories in off-policy setting, the major drawback of these algorithms is the lack of theoretical guarantees when accompanying with function approximators. It is not clear whether the algorithms converge or not, do not even mention the quality of the stationary points. That said, [182, 198] also exploit the consistency condition in Theorem 26 and propose the PCL algorithm which optimizes the upper bound of the mean squared consistency Bellman error (6.7). The same consistency condition is also discovered in [179], and the proposed Φ-learning algorithm can be viewed as fix-point iteration version of the the unified PCL with tabular Q-function. However, as we discussed in Section 6.3, the PCL algorithms becomes biased in stochastic environment, which may lead to inferior solutions [28].

Several recent works [199, 200, 201] have also considered saddle-point formulations of Bellman equations, but these formulations are fundamentally different from ours. These saddle point problems are derived from the *Lagrangian* dual of the linear programming formulation of Bellman equations [202, 203]. In contrast, our formulation is derived from the Bellman equation directly using *Fenchel* duality/transformation. It would be interesting to investigate the connection between these two saddle-point formulations in future work.

6.7 Experiments

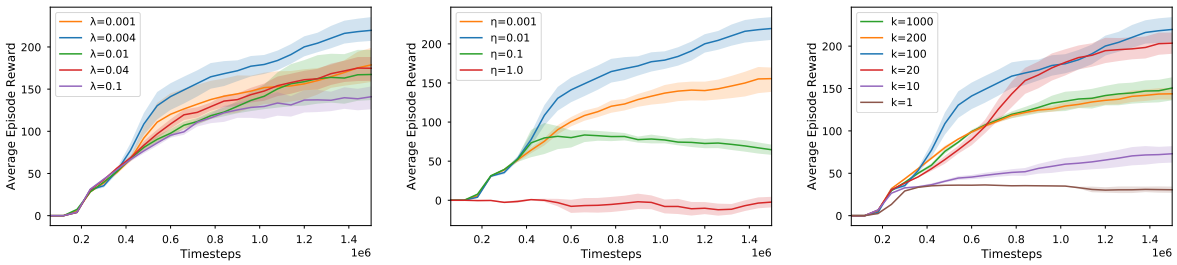
The goal of our experimental evalation is two folds: **i)**, to better understand of the effect of each algorithmic component in the proposed algorithm, and **ii)**, to demonstrate the stability and efficiency of SBEED in both *off-policy* and *on-policy* settings. Therefore, we conducted an ablation study on SBEED, and a comprehensive comparison to state-of-the-art reinforcement learning algorithms. While we derive and present SBEED for single-step Bellman error case, it can be extended to multi-step cases (Appendix D.3.2). In our experiment, we used this multi-step version.

6.7.1 Experimental Details

Policy and value function parametrization The choices of the parametrization of policy are largely based on the recent paper by [188], which shows the natural policy gradient with RBF neural network achieves the state-of-the-art performances of TRPO on MuJoCo. For the policy distribution, we parametrize it as $\pi_{\theta_\pi}(a|s) = \mathcal{N}(\mu_{\theta_\pi}(s), \Sigma_{\theta_\pi})$, where $\mu_{\theta_\pi}(s)$ is a two-layer neural nets with the random features of RBF kernel as the hidden layer and the Σ_{θ_π} is a diagonal matrix. The RBF kernel bandwidth is chosen via median trick [11, 188]. Same as [188], we use 100 hidden nodes in InvertedDoublePendulum, Swimmer, Hopper, and use 500 hidden nodes in HalfCheetah. This parametrization was used in all on-policy and off-policy algorithms for their policy functions. We adapted the linear parametrization for control variable in TRPO and Dual-AC following [201]. In DDPG and our algorithm SBEED, we need the parametrization for V and ς (or Q) as fully connected neural networks with two tanh hidden layers with 64 units each.

In the implementation of SBEED, we use the Euclidean distance for w_V and the KL-divergence for w_π in the experiments. We emphasize that other Bregman divergences are also applicable.

Training hyperparameters For all algorithms, we set $\gamma = 0.995$. All V and ς (or Q) functions of SBEED and DDPG were optimized with ADAM. The learning rates were chosen with a grid search over $\{0.1, 0.01, 0.001, 0.001\}$. For the SBEED, a stepsize of 0.005



(a) λ for entropy (b) η for dual (c) k for steps

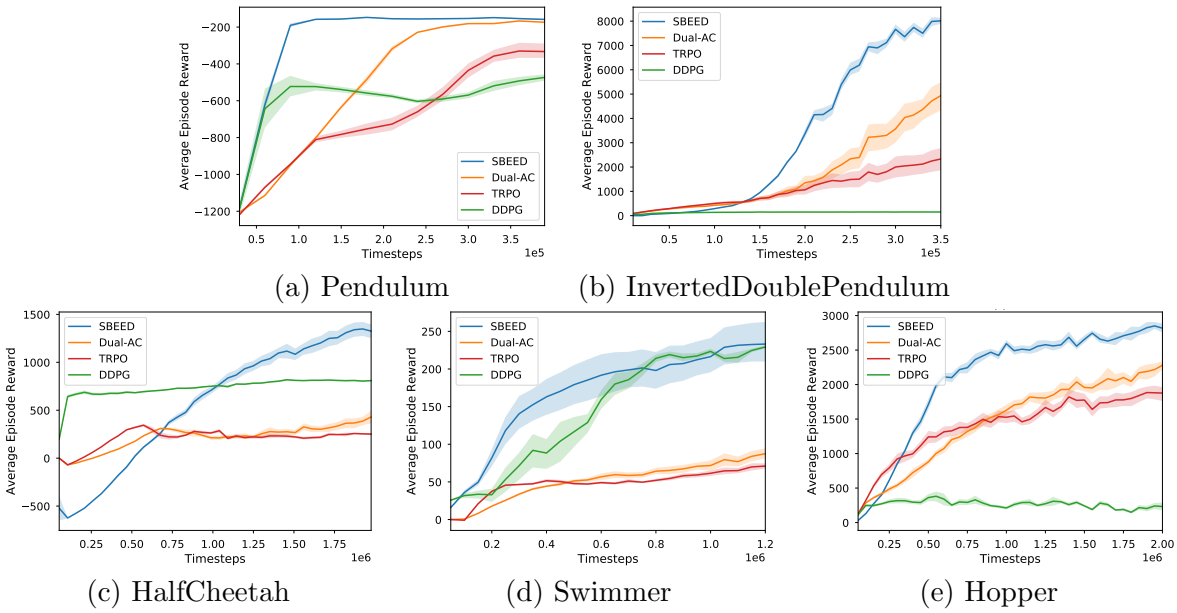
Figure 6.1: The ablation study of the SBEED on Swimmer-v1. We varied λ, η, and k to justify the effect of each component in the algorithm.

was used. For DDPG, an ADAM optimizer was also used to optimize the policy function. The learning rate is set to be $1e-4$ was used. For SBEED, η was set from a grid search of $\{0.004, 0, 01, 0.04, 0.1, 0.04\}$ and λ from a grid search in $\{0.001, 0.01, 0.1\}$. The number of the rollout steps, k was chosen by grid search from $\{1, 10, 20, 100\}$. For off-policy SBEED, a training frequency was chosen from $\{1, 2, 3\} \times 10^3$ steps. A batch size was tuned from $\{10000, 20000, 40000\}$. DDPG updated it's values every iteration and trained with a batch size tuned from $(32, 64, 128)$. For DDPG, τ was set to $1e-3$, reward scaling was set to 1, and the O-U noise σ was set to 0.3.

6.7.2 Ablation Study

To get a better understanding of the trade-off between the variance and bias, including both the bias from the smoothing technique and the introduction of the function approximator, we performed ablation study in the Swimmer-v1 environment with *stochastic* transition by varying the coefficient for entropic regularization λ and the coefficient of the dual function η in the optimization (6.10), as well as the number of the rollout steps, k.

The effect of smoothing We introduced the entropy regularization to avoid non-smoothness in Bellman error objective. However, as we discussed, it also introduces bias. We varied λ and evaluated the performance of SBEED. The results in Figure 6.1(a) are as expected: there is indeed an intermediate value for λ that gives the best bias/smoothness balance.



(a) Pendulum (b) InvertedDoublePendulum

(c) HalfCheetah (d) Swimmer (e) Hopper

Figure 6.2: The results of SBEED against TRPO, Dual AC and DDPG. Each plot shows average reward during training across 5 random runs, with 50% confidence interval. The x-axis is the number of training iterations. SBEED achieves significant better performance than the competitors on all tasks.

The effect of dual function One of the important components in our algorithm is the dual function, which cancels the variance. The effect of such a cancellation is controlled by $\eta \in [0, 1]$, and we expected an intermediate value gives the best performance. This is verified by the experiment of varying η, as shown in Figure 6.1(b).

The effect of multi-step SBEED can be easily extended to the multi-step version. However, increasing the length of steps will also increase the variance. We tested the performance of the algorithm with different k values. The results shown in Figure 6.1(c) confirms that an intermediate value for k yields the best result.

6.7.3 Comparison in Continuous Control Tasks

We tested SBEED across multiple continuous control tasks from the OpenAI Gym [204] using the MuJoCo simulator [205], including Pendulum-v0, InvertedDoublePendulum-v1, HalfCheetah-v1, Swimmer-v1, and Hopper-v1. For fairness, we follows the default setting of the MuJoCo simulator in each task in this section. These tasks have dynamics of different natures, so are helpful for evaluating the behavior of the proposed SBEED in different

scenarios. We compared SBEED with several state-of-the-art algorithms, including two on-policy algorithms, trust region policy optimization (TRPO) [186] dual actor-critic (Dual AC) [201], and one off-policy algorithm, deep deterministic policy gradient (DDPG) [206]. We did not include PCL [182] as it is a special case of our algorithm by setting $\eta = 0$, *i.e.*, ignoring the updates for dual function. Since TRPO and Dual-AC are only applicable for on-policy setting, for fairness, we also conducted the comparison with these two algorithm in on-policy setting. The results can be found in Appendix D.5.

We ran the algorithm with 5 random seeds and reported the average rewards with 50% confidence intervals. The results are shown in Figure 6.2. We can see that our SBEED algorithm achieves significantly better performance than all other algorithms across the board. These results suggest that the SBEED can exploit the off-policy samples efficiently and stably, and achieve a good trade-off between bias and variance.

It should be emphasized that the stability of algorithm is an important issue in reinforcement learning. As we can see from the results, although DDPG can also exploit the off-policy sample, which promotes its efficiency in stable environments, *e.g.*, HalfCheetah-v1 and Swimmer-v1, it may fail to learn in unstable environments, *e.g.*, InvertedDoublePendulum-v1 and Hopper-v1, which was observed by [207] and [208]. In contrast, SBEED is consistently reliable and effective in different tasks.

6.8 Summary

We reduce the policy optimizaton in MDPs to optimization with the conditional integral operator by exploiting the Nesterov's smoothing technique to Bellman optimality equation. Based on the new perspective, we develop the SBEED algorithm for policy optimization in reinforcement learning utlizing the dual embedding technique proposed in Chapter 5. The algorithm is *provably convergent* even when *nonlinear* function approximation is used on *off-policy* samples. We also provide PAC bound to characterize the sample complexity based on *one single off-policy sample path* collected by a fixed behavior policy. Empirical study shows the proposed algorithm achieves superior performance across the board, compared to state-of-the-art baselines on several MuJoCo control tasks.

CONCLUSION

In this dissertation, we propose a unified framework from the optimization with integral operators view. By considering different integral operators in the proposed framework, we obtain new perspectives for the motivated learning problems, *i.e.*, kernel methods [11], Bayesian inference [12], policy evaluation [13], and policy optimization [14], which can be categorized as learning over functions, distributions, and dynamics, respectively. Such representations of the learning problems lead to a series of novel algorithms for the corresponding problems via stochastic optimization that bypass the difficulties and drawbacks in existing algorithms:

- **Learning over functions** We reformulate most of the kernel methods as an optimization with functions as inputs through the integral operator view. Such a novel view of kernel methods avoids the explicit usage of the kernel matrices and paves the path for us to design the algorithm, which scales up the kernel machines to millions of data empirically, while still enjoys the nice theoretical properties.

- **Learning over distributions** We reformulate the Bayesian inference as an optimization with distributions as inputs on the density space through the integral operator view. Such a new view of Bayesian inference inspires us to exploit the recent advanced optimization algorithm to bypass the intractable integral in the posterior calculation. We justify the algorithm both empirically and theoretically, demonstrating the advantages of the proposed algorithm.

- **Learning over dynamics** The learning over dynamics problems is a generic class of learning problems, including invariance learning and reinforcement learning. We reformulate the problems as optimization with the conditional integral operator. With the min-max saddle-point formulation of the optimization, we design the stochastic algorithm which only needs the sample from joint distribution and avoids the difficulty in estimating the conditional integral due to lack of samples. We provide the

theoretical analysis of the algorithm and apply the algorithm for invariance learning, policy evaluation. We then investigate the policy optimization with the developed technique, resulted a provably convergent algorithm even when nonlinear function approximation is used on off-policy samples, what we believe to be the first.

As we demonstrated in the thesis, the proposed framework provides us the relatively abstract perspective to unify the learning problems, while still preserves the useful properties through the structure of the integral operator for the algorithm design. For the future work, our research will continue to explore other important learning problems via the integral operator view for better algorithms. For instance:

Undirected graphical model and exponential families The infinite dimensional exponential families [209] is defined as, $p(x) = p_0(x)\exp(f(x) - A(f))$, within $\mathcal{F} := \{f \in \mathcal{H} : \exp(A(f)) < \infty\}$, $A(f) := \log \int_{\mathcal{X}} \exp(f(x)) p_0(x)\, dx$ and $\mathcal{H}$ denotes some RKHS. Since the integral in $A(f)$ is intractable in maximum-likelihood (MLE), [209] introduce the score matching to fit f, which is not scalable due to the computation and storage of the kernel matrices and its derivatives.

Based on the integral operator view of MLE, we can easily obtain a scalable algorithm for general exponential families estimation by combining doubly stochastic gradients and dual embedding. Specifically, by dual embedding, we can reformulate the objective of MLE, $\max_{f\in\mathcal{H}} \widehat{\mathbb{E}}_x[f(x)] - A(f)$, as

$$\max_{g\in L_2(\rho)} \min_{q\in\mathcal{P}} \widehat{\mathbb{E}}_x[(\mathcal{Q}g)(x)] - \mathbb{E}_{x\sim q(x)}[(\mathcal{Q}g)(x)] + KL(q||p_0), \tag{6.11}$$

where the $f \in \mathcal{H}$ is represented by $\mathcal{Q}g$ for scalability. The optimization (6.11) is convex-concave w.r.t. f and q. Then, we can apply the doubly stochastic gradient descent to the above optimization (6.11).

What follows naturally and logically is to extend this view to design algorithms for exponential families estimation with structures in potential functions, *e.g.*, $f(x)$ can be decomposed into cliques or $f(x) = \int h(x, z)\, dz$. Due to the importance and generality of the exponential families, such a problem is significant in machine learning. In this scenario, the

first question is how to represent the structure information in the dual parametrization, *i.e.*, $q \in \mathcal{P}$. The second question is once the convexity-concavity structure in the optimization does no longer hold due to the structure constraints introduced, how to perform learning while still keep the desired properties. It is also interesting to address the connections between the algorithm and GANs [210] and contrastive divergence [211].

Meta-learning The meta-learning problems focus on learning the components to design algorithms, *e.g.*, the update rules [212, 213, 214] and the initialization [215], which can be applied to downstream learning tasks. The integral operator view unifies the meta-learning problems as *learning over algorithms*. Specifically, denote the initialization as f and the algorithm update as $\mathcal{U}$, then, after t-iteration, the algorithm outputs $\mathcal{U}^t f$ where $\mathcal{U}^t := \underbrace{\mathcal{U} \circ \mathcal{U} \circ \ldots \circ \mathcal{U}}_{t}$, and the learning problem is reformulated as

$$\min_{\mathcal{U}, f} \ \mathbb{E}_D \left[J \left(\mathcal{U}^t f, D \right) \right] + \nu G \left(\mathcal{U}, f \right), \tag{6.12}$$

where D denotes the tasks and J can be the evaluation criteria for the algorithm. Actually, the optimization (6.12) can be applied for learning to sample [216] with $\mathcal{U}$ as the transition kernel and f as the proposal distribution. $J(\cdot, \cdot)$ can be designed to match the target distribution and $\mathcal{U}^t f$ and to accelerate the convergence of the sampling algorithm.

The difficulty of the optimization (6.12) lies on both theoretical and practical aspects. Theoretically, the objective (6.12) is not convex w.r.t. $(\mathcal{U}, f)$ jointly. The parametrization of $\mathcal{U}$ and f will introduce extra non-convexity. What guarantee we can obtain about $(\mathcal{U}, f)$, and how its performance compared with the hand-designed algorithm theoretically are interesting. Practically, how to implement the stochastic optimization in operator and function space should be carefully designed to achieve scalability and efficiency. In our future study, we would like to address these issues and make the algorithm practical.

Given the successes of the framework, we believe the proposed unified framework provides new directions for machine learning research and can be potentially useful to design new algorithms for many other scenarios beyond the representative learning problems discussed in this dissertation.

Appendices

APPENDIX A
PROOF DETAILS IN CHAPTER 3

A.1 Convergence Rate

We first provide specific bounds and detailed proofs for the two error terms appeared in Theorem 11 and Theorem 12.

A.1.1 Error due to random features

Lemma 31 *We have*

(i) For any $x \in \mathcal{X}$, $\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[|f_{t+1}(x) - h_{t+1}(x)|^2] \leqslant B_{1,t+1}^2 := 4M^2(\kappa+\phi)^2 \sum_{i=1}^t |a_t^i|^2$.

(ii) For any $x \in \mathcal{X}$, *with probability at least* $1-\delta$ *over* $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,

$$|f_{t+1}(x) - h_{t+1}(x)|^2 \leqslant B_{2,t+1}^2 := 2M^2(\kappa+\phi)^2 \ln\left(\frac{2}{\delta}\right) \sum_{i=1}^t |a_t^i|^2$$

Proof Let $V_i(x) = V_i(x; \mathcal{D}^i, \boldsymbol{\omega}^i) := a_t^i \left(\zeta_i(x) - \xi_i(x)\right)$. Since $V_i(x)$ is a function of $(\mathcal{D}^i, \boldsymbol{\omega}^i)$ and

$$\mathbb{E}_{\mathcal{D}^i,\boldsymbol{\omega}^i}\left[V_i(x)|\boldsymbol{\omega}^{i-1}\right] = a_t^i \mathbb{E}_{\mathcal{D}^i,\boldsymbol{\omega}^i}\left[\zeta_i(x) - \xi_i(x)|\boldsymbol{\omega}^{i-1}\right] = a_t^i \mathbb{E}_{\mathcal{D}^i,\boldsymbol{\omega}^{i-1}}\left[\mathbb{E}_{\omega^i}\left[\zeta_i(x) - \xi_i(x)|\boldsymbol{\omega}^{i-1}\right]\right] = 0,$$

we have that $\{V_i(x)\}$ is a martingal difference sequence. Further note that

$$|V_i(x)| \leqslant c_i = 2M(\phi+\kappa)|a_t^i|.$$

Then by Azuma's Inequality, for any $\epsilon > 0$,

$$\Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{|\sum_{i=1}^t V_i(x)| \geqslant \epsilon\right\} \leqslant 2\exp\left\{-\frac{2\epsilon^2}{\sum_{i=1}^t c_i^2}\right\}$$

which is equivalent as

$$\Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{\left(\sum_{i=1}^t V_i(x)\right)^2 \geqslant \ln(2/\delta)\sum_{i=1}^t c_i^2/2\right\} \leqslant \delta.$$

Moreover,

$$\begin{aligned}\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left[\left(\sum_{i=1}^t V_i(x)\right)^2\right] &= \int_0^\infty \Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{\left(\sum_{i=1}^t V_i(x)\right)^2 \geqslant \epsilon\right\} d\epsilon \\ &= \int_0^\infty 2\exp\left\{-\frac{2\epsilon}{\sum_{i=1}^t c_i^2}\right\} d\epsilon = \sum_{i=1}^t c_i^2.\end{aligned}$$

Since $f_{t+1}(x) - h_{t+1}(x) = \sum_{i=1}^t V_i(x)$, we immediately obtain the two parts of the lemma. ■

Lemma 32 *Suppose $\gamma_i = \frac{\theta}{i}(1 \leqslant i \leqslant t)$ and $\theta\nu \in (1,2) \cup \mathbb{Z}_+$. Then we have*

(1) $|a_t^i| \leqslant \frac{\theta}{t}$. Consequently, $\sum_{i=1}^t (a_t^i)^2 \leqslant \frac{\theta^2}{t}$.

(2) $$\sum_{i=1}^t \gamma_i |a_t^i| \leqslant \begin{cases} \frac{\theta^2(\ln(t)+1)}{t}, & \text{if } \theta\nu \in [1,2), \\ \frac{\theta^2}{t}, & \text{if } \theta\nu \in [2,+\infty) \cap \mathbb{Z}_+ \end{cases}.$$

Proof (1) follows by induction on i. $|a_t^t| \leqslant \frac{\theta}{t}$ is trivially true. We have

$$|a_t^i| = |a_t^{i+1}\frac{\gamma_i}{\gamma_{i+1}}(1-\nu\gamma_{i+1})| = \frac{i+1}{i}|1 - \frac{\nu\theta}{i+1}| \cdot |a_t^{i+1}| = |\frac{i+1-\nu\theta}{i}| \cdot |a_t^{i+1}|.$$

When $\nu\theta \in (1,2)$, $i - 1 < i + 1 - \nu\theta < i$ for any $i \geqslant 1$, so $|a_t^i| < |a_t^{i+1}| \leqslant \frac{\theta}{t}$. When $\nu\theta \in \mathbb{Z}_+$, if $i > \nu\theta - 1$, then $|a_t^i| < |a_t^{i+1}| \leqslant \frac{\theta}{t}$; if $i \leqslant \nu\theta - 1$, then $|a_t^i| = 0$. For (2), when $\theta\nu \in [1,2)$,

$$\sum_{i=1}^t \gamma_t |a_t^i| = \sum_{i=1}^t \frac{\theta^2}{i^2} \cdot \frac{i+1-\theta\nu}{i+1} \cdots \frac{t-\theta\nu}{t} \leqslant \sum_{i=1}^t \frac{\theta^2}{i^2} \cdot \frac{i}{i+1} \cdots \frac{t-1}{t} \leqslant \sum_{i=1}^t \frac{\theta^2}{it} \leqslant \frac{\theta^2(\ln(t)+1)}{t}.$$

When $\theta\nu \in \mathbb{Z}_+$ and $2 \leqslant \theta\nu \leqslant t$,

$$\sum_{i=1}^t \gamma_t |a_t^i| = \sum_{i=2}^t \frac{\theta^2}{i^2} \cdot \frac{i+1-\theta\nu}{i+1} \cdots \frac{t-\theta\nu}{t} \leqslant \sum_{i=1}^t \frac{\theta^2}{i^2} \cdot \frac{i-1}{i+1} \cdots \frac{t-2}{t} \leqslant \sum_{i=2}^t \frac{\theta^2(i-1)}{it(t-1)} \leqslant \frac{\theta^2}{t}.$$

■

A.1.2 Error due to random data

Lemma 33 *Assume $\ell'(u,y)$ is L-Lipschitz continous in terms of $u \in \mathbb{R}$. Let f^* be the optimal solution to our target problem. Then*

(i) If we set $\gamma_t = \frac{\theta}{t}$ with θ such that $\theta\nu \in (1,2) \cup \mathbb{Z}_+$, then

$$\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}\left[\|h_{t+1} - f^*\|_{\mathcal{H}}^2\right] \leqslant \frac{Q_1^2}{t},$$

where

$$Q_1 = \max\left\{\|f^*\|_{\mathcal{H}}, \frac{Q_0 + \sqrt{Q_0^2 + (2\theta\nu - 1)(1+\theta\nu)^2\theta^2\kappa M^2}}{2\nu\theta - 1}\right\},$$

$$Q_0 = 2\sqrt{2}\kappa^{1/2}(\kappa + \phi)LM\theta^2.$$

Particularly, if $\theta\nu = 1$, we have $Q_1 \leqslant \max\left\{\|f^*\|_{\mathcal{H}}, 4\sqrt{2}((\kappa+\phi)L + \nu) \cdot \frac{\kappa^{1/2}M}{\nu^2}\right\}$.

(ii) If we set $\gamma_t = \frac{\theta}{t}$ with θ such that $\theta\nu \in \mathbb{Z}_+$ and $t \geqslant \theta\nu$, then with probability at least $1 - 2\delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,

$$\|h_{t+1} - f^*\|_{\mathcal{H}}^2 \leqslant Q_2^2 \frac{\ln(2t/\delta)\ln(t)}{t}.$$

where

$$Q_2 = \max\left\{\|f^*\|_{\mathcal{H}}, Q_0 + \sqrt{Q_0^2 + \kappa M^2(1+\theta\nu)^2(\theta^2 + 16\theta/\nu)}\right\},$$

$$Q_0 = 4\sqrt{2}\kappa^{1/2}M\theta(8 + (\kappa+\phi)\theta L).$$

Particularly, if $\theta\nu = 1$, we have $Q_2 \leqslant \max\left\{\|f^*\|_{\mathcal{H}}, 8\sqrt{2}((\kappa+\phi)L + 9\nu) \cdot \frac{\kappa^{1/2}M}{\nu^2}\right\}$.

Proof For the sake of simple notations, let us first denote the following three different gradient terms, which are

$$g_t = \xi_t + \nu h_t = \ell'(f_t(x_t), y_t)k(x_t, \cdot) + \nu h_t,$$
$$\hat{g}_t = \hat{\xi}_t + \nu h_t = \ell'(h_t(x_t), y_t)k(x_t, \cdot) + \nu h_t,$$
$$\bar{g}_t = \mathbb{E}_{\mathcal{D}_t}[\hat{g}_t] = \mathbb{E}_{\mathcal{D}_t}\left[\ell'(h_t(x_t), y_t)k(x_t, \cdot)\right] + \nu h_t.$$

Note that by our previous definition, we have $h_{t+1} = h_t - \gamma_t g_t, \forall t \geqslant 1$.

Denote $A_t = \|h_t - f^*\|_{\mathcal{H}}^2$. Then we have

$$\begin{aligned} A_{t+1} &= \|h_t - f^* - \gamma_t g_t\|_{\mathcal{H}}^2 \\ &= A_t + \gamma_t^2 \|g_t\|_{\mathcal{H}}^2 - 2\gamma_t \langle h_t - f^*, g_t \rangle_{\mathcal{H}} \\ &= A_t + \gamma_t^2 \|g_t\|_{\mathcal{H}}^2 - 2\gamma_t \langle h_t - f^*, \bar{g}_t \rangle_{\mathcal{H}} + 2\gamma_t \langle h_t - f^*, \bar{g}_t - \hat{g}_t \rangle_{\mathcal{H}} + 2\gamma_t \langle h_t - f^*, \hat{g}_t - g_t \rangle_{\mathcal{H}} \end{aligned}$$

Because of the strongly convexity of (3.2) and optimality condition, we have

$$\langle h_t - f^*, \bar{g}_t \rangle_{\mathcal{H}} \geqslant \nu \|h_t - f^*\|_{\mathcal{H}}^2$$

Hence, we have

$$A_{t+1} \leqslant (1 - 2\gamma_t \nu) A_t + \gamma_t^2 \|g_t\|_{\mathcal{H}}^2 + 2\gamma_t \langle h_t - f^*, \bar{g}_t - \hat{g}_t \rangle_{\mathcal{H}} + 2\gamma_t \langle h_t - f^*, \hat{g}_t - g_t \rangle_{\mathcal{H}}, \forall t \geqslant 1 \quad \text{(A.1)}$$

Proof for (i): Let us denote $\mathcal{M}_t = \|g_t\|_{\mathcal{H}}^2, \mathcal{N}_t = \langle h_t - f^*, \bar{g}_t - \hat{g}_t \rangle_{\mathcal{H}}, \mathcal{R}_t = \langle h_t - f^*, \hat{g}_t - g_t \rangle_{\mathcal{H}}$. We first show that $\mathcal{M}_t, \mathcal{N}_t, \mathcal{R}_t$ are bounded. Specifically, we have for $t \geqslant 1$,

(1) $\mathcal{M}_t \leqslant \kappa M^2 (1 + \nu c_t)^2$, where $c_t = \sqrt{\sum_{i,j=1}^{t-1} |a_{t-1}^i| \cdot |a_{t-1}^j|}$ for $t \geqslant 2$ and $c_1 = 0$;

(2) $\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}[\mathcal{N}_t] = 0$;

(3) $\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}[\mathcal{R}_t] \leqslant \kappa^{1/2} L B_{1,t} \sqrt{\mathbb{E}_{\mathcal{D}^{t-1}, \boldsymbol{\omega}^{t-1}}[A_t]}$, where $B_{1,t}^2 := 4M^2(\kappa + \phi)^2 \sum_{i=1}^{t-1} |a_{t-1}^i|^2$ for $t \geqslant 2$ and $B_{1,1} = 0$;

We prove these results separately in Lemma 34 below. Let us denote $e_t = \mathbb{E}_{\mathcal{D}^{t-1}, \boldsymbol{\omega}^{t-1}}[A_t]$, given the above bounds, we arrive at the following recursion,

$$e_{t+1} \leqslant (1 - 2\gamma_t \nu) e_t + \kappa M^2 \gamma_t^2 (1 + \nu c_t)^2 + 2\kappa^{1/2} L \gamma_t B_{1,t} \sqrt{e_t}. \quad \text{(A.2)}$$

When $\gamma_t = \theta / t$ with θ such that $\theta \nu \in (1, 2) \cup \mathbb{Z}_+$, from Lemma 32, we have $|a_t^i| \leqslant \frac{\theta}{t}, \forall 1 \leqslant i \leqslant t$. Consequently, $c_t \leqslant \theta$ and $B_{1,t}^2 \leqslant 4M^2(\kappa + \phi) \frac{\theta^2}{t-1}$. Applying these bounds leads to the

refined recursion as follows

$$e_{t+1} \leqslant \left(1 - \frac{2\nu\theta}{t}\right) e_t + \kappa M^2 \frac{\theta^2}{t^2}(1+\nu\theta)^2 + 2\kappa^{1/2} L \frac{\theta}{t} \sqrt{4M^2(\kappa+\phi)^2 \frac{\theta^2}{t-1}} \sqrt{e_t}$$

that can be further written as

$$e_{t+1} \leqslant \left(1 - \frac{2\nu\theta}{t}\right) e_t + \frac{\beta_1}{t}\sqrt{\frac{e_t}{t}} + \frac{\beta_2}{t^2},$$

where $\beta_1 = 4\sqrt{2}\kappa^{1/2} LM(k+\phi)\theta^2$ and $\beta_2 = \kappa M^2(1+\nu\theta)^2\theta^2$. Invoking Lemma 38 with $\eta = 2\theta\nu > 1$, we obtain

$$e_t \leqslant \frac{Q_1^2}{t},$$

where $Q_1 = \max\left\{\|f^*\|_{\mathcal{H}}, \frac{Q_0 + \sqrt{Q_0^2 + (2\theta\nu-1)(1+\theta\nu)^2\theta^2\kappa M^2}}{2\nu\theta - 1}\right\}$, and $Q_0 = 2\sqrt{2}\kappa^{1/2}(\kappa+\phi)LM\theta^2$.

Proof for (ii): Cumulating equations (A.1) with $i = 1, \ldots t$, we end up with the following inequality

$$\begin{aligned} A_{t+1} \leqslant & \textstyle\prod_{i=1}^t (1-2\gamma_i\nu) A_1 + 2\sum_{i=1}^t \gamma_i \prod_{j=i+1}^t (1-2\nu\gamma_j)\langle h_i - f^*, \bar{g}_i - \hat{g}_i\rangle_{\mathcal{H}} \\ & + 2\sum_{i=1}^t \gamma_i \prod_{j=i+1}^t (1-2\nu\gamma_j)\langle h_i - f^*, \hat{g}_i - g_i\rangle_{\mathcal{H}} + \sum_{i=1}^t \gamma_i^2 \prod_{j=i+1}^t (1-2\nu\gamma_j)\,\|g_i\|_{\mathcal{H}}^2 \end{aligned} \tag{A.3}$$

Let us denote $b_t^i = \gamma_i \prod_{j=i+1}^t (1-2\nu\gamma_j), 1 \leqslant i \leqslant t$, the above inequality is equivalent as

$$A_{t+1} \leqslant \prod_{i=1}^t (1-2\gamma_i\nu) A_1 + \sum_{i=1}^t \gamma_i b_t^i \mathcal{M}_i + 2\sum_{i=1}^t b_t^i \mathcal{N}_i + 2\sum_{i=1}^t b_t^i \mathcal{R}_i$$

We first show that

(4) for any $0 < \delta < 1/e$ and $t \geqslant 4$, with probability $1-\delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,

$$\textstyle\sum_{i=1}^t b_t^i \mathcal{N}_i \leqslant 2\max\left\{4\kappa^{1/2} M \sqrt{\sum_{i=1}^t (b_t^i)^2 A_i},\ \max_i |b_t^i| \cdot C_0 \sqrt{\ln(\ln(t)/\delta)}\right\} \sqrt{\ln(\ln(t)/\delta)},$$

where $C_0 = \frac{4\max_{1\leqslant i\leqslant t}\mathcal{M}_i}{\nu}$.

(5) for any $\delta > 0$, with probability $1-\delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,

$$\textstyle\sum_{i=1}^t b_t^i \mathcal{R}_i \leqslant \sum_{i=1}^t b_t^i \kappa^{1/2} L \hat{B}_{2,i} \sqrt{A_i},$$

where $\hat{B}_{2,i}^2 = 2M^2(\kappa+\phi)^2 \ln\left(\frac{2t}{\delta}\right) \sum_{j=1}^{i-1} |a_{i-1}^j|^2$.

Again, the proofs of these results are given separately in Lemma 34. Applying the above bounds leads to the refined recursion as follows,

$$\begin{aligned} A_{t+1} \quad \leqslant \quad & \prod_{i=1}^t (1-2\gamma_i\nu) A_1 + \sum_{i=1}^t \gamma_i b_t^i \mathcal{M}_i + 2\sum_{i=1}^t b_t^i \kappa^{1/2} L B_{2,i} \sqrt{A_i} \\ & + 4\max\left\{ 4\kappa^{1/2} M \sqrt{\sum_{i=1}^t (b_t^i)^2 A_i},\ \max_i |b_t^i| \cdot C_0 \sqrt{\ln(\ln(t)/\delta)} \right\} \sqrt{\ln(\ln(t)/\delta)} \end{aligned}$$

with probability $1-2\delta$. When $\gamma_t = \theta/t$ with θ such that $\theta\nu \in \mathbb{Z}_+$, with similar reasons in Lemma 32, we have $|b_t^i| \leqslant \frac{\theta}{t}, 1 \leqslant i \leqslant t$ and also we have $\prod_{i=1}^t (1-2\gamma_i\nu) = \prod_{i=1}^{\theta\nu-1}(1-2\frac{\theta\nu}{i}) \prod_{i=\theta\nu+1}^t (1-2\frac{\theta\nu}{i})(1-2\frac{\theta\nu}{\theta\nu}) = 0$, and $\sum_{i=1}^t \gamma_i b_t^i \leqslant \frac{\theta^2}{t}$. Therefore, we can rewrite the above recursion as

$$A_{t+1} \leqslant \frac{\beta_1}{t} + \beta_2 \sqrt{\ln(2t/\delta)} \cdot \sum_{i=1}^t \frac{\sqrt{A_i}}{t\sqrt{i}} + \beta_3 \sqrt{\ln(\ln(t)/\delta)} \frac{\sqrt{\sum_{i=1}^t A_i}}{t} + \beta_4 \ln(\ln(t/\delta)) \frac{1}{t} \quad \text{(A.4)}$$

where $\beta_1 = \kappa M^2 (1+\nu\theta)^2 \theta^2$, $\beta_2 = 2\sqrt{2}\kappa^{1/2} L M (\kappa+\phi)\theta^2$, $\beta_3 = 16\kappa^{1/2} M\theta$, $\beta_4 = 16\kappa M^2 (1+\theta\nu)^2 \theta/\nu$. Invoking Lemma 39, we obtain

$$A_{t+1} \leqslant \frac{Q_2^2 \ln(2t/\delta) \ln^2(t)}{t},$$

with the specified Q_2. ■

Lemma 34 *In this lemma, we prove the inequalities (1)–(5) in Lemma 33.*

Proof Given the definitions of $\mathcal{M}_t, \mathcal{N}_t, \mathcal{R}_t$ in Lemma 33, we have

(1) $\mathcal{M}_t \leqslant \kappa M^2 (1 + \nu \sqrt{\sum_{i,j=1}^{t-1} |a_{t-1}^i| \cdot |a_{t-1}^j|})^2$;

This is because

$$\mathcal{M}_t = \|g_t\|_{\mathcal{H}}^2 = \|\xi_t + \nu h_t\|_{\mathcal{H}}^2 \leqslant (\|\xi_t\|_{\mathcal{H}} + \nu\|h_t\|_{\mathcal{H}})^2.$$

We have

$$\|\xi_t\|_{\mathcal{H}} = \|\ell'(f_t(x_t), y_t)k(x_t, \cdot)\|_{\mathcal{H}} \leqslant \kappa^{1/2} M,$$

and

$$\begin{aligned}\|h_t\|_{\mathcal{H}}^2 &= \sum_{i=1}^{t-1}\sum_{j=1}^{t-1} a_{t-1}^i a_{t-1}^j \ell'(f_i(x_i), y_i)\ell'(f_j(x_j), y_j)k(x_i, x_j) \\ &\leqslant \kappa M^2 \sum_{i=1}^{t-1}\sum_{j=1}^{t-1} |a_{t-1}^i| \cdot |a_{t-1}^j|.\end{aligned}$$

(2) $\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}[\mathcal{N}_t] = 0$;

This is because $\mathcal{N}_t = \langle h_t - f^*, \bar{g}_t - \hat{g}_t\rangle_{\mathcal{H}}$,

$$\begin{aligned}\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}[\mathcal{N}_t] &= \mathbb{E}_{\mathcal{D}^{t-1}, \boldsymbol{\omega}^t}\left[\mathbb{E}_{D_t}\left[\langle h_t - f^*, \bar{g}_t - \hat{g}_t\rangle_{\mathcal{H}} | \mathcal{D}^{t-1}, \boldsymbol{\omega}^t\right]\right] \\ &= \mathbb{E}_{\mathcal{D}^{t-1}, \boldsymbol{\omega}^t}\left[\langle h_t - f^*, \mathbb{E}_{D_t}\left[\bar{g}_t - \hat{g}_t\right]\rangle_{\mathcal{H}}\right] \\ &= 0.\end{aligned}$$

(3) $\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t}[\mathcal{R}_t] \leqslant \kappa^{1/2} L B_{1,t} \sqrt{\mathbb{E}_{\mathcal{D}^{t-1}, \boldsymbol{\omega}^{t-1}}[A_t]}$, where $B_{1,t}^2 := 4M^2(\kappa + \phi)^2 \sum_{i=1}^{t-1} |a_{t-1}^i|^2$;

This is because $\mathcal{R}_t = \langle h_t - f^*, \hat{g}_t - g_t \rangle_{\mathcal{H}}$,

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[\mathcal{R}_t] &= \mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left[\langle h_t - f^*, \hat{g}_t - g_t \rangle_{\mathcal{H}}\right] \\
&= \mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left[\langle h_t - f^*, [\ell'(f_t(x_t), y_t) - \ell'(h_t(x_t), y_t)]k(x_t, \cdot)\rangle_{\mathcal{H}}\right] \\
&\leqslant \mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left[|\ell'(f_t(x_t), y_t) - \ell'(h_t(x_t), y_t)| \cdot \|k(x_t, \cdot)\|_{\mathcal{H}} \cdot \|h_t - f^*\|_{\mathcal{H}}\right] \\
&\leqslant \kappa^{1/2} L \cdot \mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left[|f_t(x_t) - h_t(x_t)| \, \|h_t - f^*\|_{\mathcal{H}}\right] \\
&\leqslant \kappa^{1/2} L \sqrt{\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}|f_t(x_t) - h_t(x_t)|^2} \sqrt{\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\|h_t - f^*\|_{\mathcal{H}}^2} \\
&\leqslant \kappa^{1/2} L B_{1,t} \sqrt{\mathbb{E}_{\mathcal{D}^{t-1},\boldsymbol{\omega}^{t-1}}[A_t]}
\end{aligned}$$

where the first and third inequalities are due to Cauchy–Schwarz Inequality and the second inequality is due to L-Lipschitz continuity of $\ell'(\cdot, \cdot)$ in the first parameter, and the last step is due to Lemma 31 and the definition of A_t.

(4) for any $0 < \delta < 1/e$ and $t \geqslant 4$, with probability at least $1 - \delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,

$$\textstyle\sum_{i=1}^t b_t^i \mathcal{N}_i \leqslant 2 \max\left\{4\kappa^{1/2} M \sqrt{\sum_{i=1}^t (b_t^i)^2 A_i},\ \max_i |b_t^i| \cdot C_0 \sqrt{\ln(\ln(t)/\delta)}\right\} \sqrt{\ln(\ln(t)/\delta)},$$

where $C_0 = \frac{4 \max_{1 \leqslant i \leqslant t} \mathcal{M}_i}{\nu}$.

This result follows directly from Lemma 3 in [217]. Let us define $d_i = d_i(\mathcal{D}^i, \boldsymbol{\omega}^i) := b_t^i \mathcal{N}_i = b_t^i \langle h_i - f^*, \bar{g}_i - \hat{g}_i \rangle_{\mathcal{H}}, 1 \leqslant i \leqslant t$, we have

- $\{d_i\}_{i=1}^t$ is martingale difference sequence since $\mathbb{E}_{\mathcal{D}^i,\boldsymbol{\omega}^i}\left[\mathcal{N}_i | \mathcal{D}^{i-1}, \boldsymbol{\omega}^{i-1}\right] = 0$.
- $|d_i| \leqslant \max_i |b_t^i| \cdot C_0$, with $C_0 = \frac{4 \max_{1 \leqslant i \leqslant t} \mathcal{M}_i}{\nu}$, $\forall 1 \leqslant i \leqslant t$.
- $Var(d_i | \mathcal{D}^{i-1}, \boldsymbol{\omega}^{i-1}) \leqslant 4\kappa M^2 |b_t^i|^2 A_i, \forall 1 \leqslant i \leqslant t$.

Plugging in these specific bounds in Lemma 3 in [Alexander et.al., 2012], which is,

$$\Pr\left(\textstyle\sum_{i=1}^t d_t \geqslant 2 \max\{2\sigma_t, d_{max} \sqrt{\ln(1/\delta)}\} \sqrt{\ln(1/\delta)}\right) \leqslant \ln(t)\delta.$$

where $\sigma_t^2 = \sum_{i=1}^t Var_{i-1}(d_i)$ and $d_{max} = \max_{1 \leqslant i \leqslant t} |d_i|$, we immediately obtain the above inequality as desired.

(5) for any $\delta > 0$, with probability at least $1 - \delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$,

$$\sum_{i=1}^t b_t^i \mathcal{R}_i \leqslant \sum_{i=1}^t |b_t^i| \kappa^{1/2} L \hat{B}_{2,i} \sqrt{A_i},$$

where $\hat{B}_{2,i}^2 = 2M^2(\kappa+\phi)^2 \ln\left(\frac{2t}{\delta}\right) \sum_{j=1}^{i-1} |a_{i-1}^j|^2$.

This is because, for any $1 \leqslant i \leqslant t$, recall that from analysis in (3), we have $\mathcal{R}_i \leqslant \kappa^{1/2} L |f_t(x_t) - h_t(x_t)| \cdot \|h_t - f^*\|_{\mathcal{H}}$, therefore from Lemma 33,

$$\Pr(b_t^i \mathcal{R}_i \leqslant \kappa^{1/2} L |b_t^i| \hat{B}_{2,i} \sqrt{A_i}) \geqslant \Pr(|f_i(x_i) - h_i(x_i)|^2 \leqslant \hat{B}_{2,i}^2) \geqslant 1 - \delta/t.$$

Taking the sum over i, we therefore get

$$\Pr(\textstyle\sum_{i=1}^t b_t^i \mathcal{R}_i \leqslant \sum_{i=1}^t |b_t^i| \kappa^{1/2} L B_{2,i} \sqrt{A_i}) \geqslant 1 - \delta.$$

■

Applying these lemmas immediately gives us Theorem 11 and Theorem 12, which implies pointwise distance between the solution $f_{t+1}(\cdot)$ and $f^*(\cdot)$. Now we prove similar bounds in the sense of L_∞ and L_2 distance.

A.2 L_∞ distance, L_2 distance, and generalization bound

Corollary 35 (L_∞ distance) *Theorem 11 also implies a bound in L_∞ sense, namely,*

$$\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t} \|f_{t+1} - f^*\|_\infty^2 \leqslant \frac{2C^2 + 2\kappa Q_1^2}{t}.$$

Consequently, for the average solution $\hat{f}_{t+1}(\cdot) := \frac{1}{t}\sum_{i=1}^t f_i(\cdot)$, we also have

$$\mathbb{E}_{\mathcal{D}^t, \boldsymbol{\omega}^t} \|\hat{f}_{t+1} - f^*\|_\infty^2 \leqslant \frac{(2C^2 + 2\kappa Q_1^2)(\ln(t)+1)}{t}.$$

This is because $\|f_{t+1} - f^*\|_\infty = \max_{x\in\mathcal{X}} |f_{t+1}(x) - f^*(x)| = |f_{t+1}(x_*) - f^*(x_*)|$, where $x_* \in \mathcal{X}$ always exists since $\mathcal{X}$ is closed and bounded. Note that the result for average solution can be improved without log factor using more sophisticated analysis (see also reference in [217]).

Corollary 36 (L_2 distance) *With the choices of γ_t in Lemma 33, we have*

(i) $\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}\|f_{t+1} - f^*\|_2^2 \leqslant \frac{2C^2+2\kappa Q_1^2}{t}$,

(ii) $\|f_{t+1} - f^*\|_2^2 \leqslant \frac{C^2\ln(8\sqrt{e}t/\delta)+2\kappa Q_2^2\ln(2t/\delta)\ln^2(t)}{t}$, *with probability at least* $1-3\delta$ *over* $(\mathcal{D}^t, \boldsymbol{\omega}^t)$.

Proof *(i)* follows directly from Theorem 11. *(ii)* can be proved as follows. First, we have

$$\|f_{t+1} - f^*\|_2^2 = \mathbb{E}_x|f_{t+1}(x) - f^*(x)|^2 \leqslant 2\mathbb{E}_x|f_{t+1}(x) - h_{t+1}(x)|^2 + 2\kappa\|h_{t+1} - f^*\|_{\mathcal{H}}.$$

From Lemma 33, with probability at least $1-2\delta$, we have

$$\|h_{t+1} - f^*\|_{\mathcal{H}}^2 \leqslant \frac{Q_2^2\ln(2t/\delta)\ln^2(t)}{t}. \tag{A.5}$$

From Lemma 31, for any $x \in \mathcal{X}$, we have

$$\Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{|f_{t+1}(x) - h_{t+1}(x)|^2 \geqslant \frac{2(\kappa+\phi)^2M^2\ln(\frac{2}{\epsilon})\theta^2}{t}\right\} \leqslant \epsilon.$$

Since $C^2 = 4(\kappa+\phi)^2M^2\theta^2$, the above inequality can be writen as

$$\Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{|f_{t+1}(x) - h_{t+1}(x)|^2 \geqslant \frac{C^2\ln(\frac{2}{\epsilon})}{2t}\right\} \leqslant \epsilon.$$

which leads to

$$\Pr_{x\sim p(x)}\Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{|f_{t+1}(x) - h_{t+1}(x)|^2 \geqslant \frac{C^2\ln(\frac{2}{\epsilon})}{2t}\right\} \leqslant \epsilon.$$

By Fubini's theorem and Markov's inequality, we have

$$\Pr_{\mathcal{D}^t,\boldsymbol{\omega}^t}\left\{\Pr_{x\sim p(x)}\left\{|f_{t+1}(x) - h_{t+1}(x)|^2 \geqslant \frac{C^2\ln(\frac{2}{\epsilon})}{2t}\right\} \geqslant \frac{\epsilon}{\delta}\right\} \leqslant \delta.$$

From the analysis in Lemma 31, we also have that $|f_{t+1}(x) - h_{t+1}(x)| \leqslant C^2$. Therefore, with probability at least $1-\delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$, we have

$$\mathbb{E}_{x\sim p(x)}[|f_{t+1}(x) - h_{t+1}(x)|^2] \leqslant \frac{C^2 \ln(\frac{2}{\epsilon})}{2t}(1 - \frac{\epsilon}{\delta}) + C^2\frac{\epsilon}{\delta}$$

Let $\epsilon = \frac{\delta}{4t}$, we have

$$\mathbb{E}_{x\sim p(x)}[|f_{t+1}(x) - h_{t+1}(x)|^2] \leqslant \frac{C^2}{2t}(\ln(8t/\delta) + \frac{1}{2}) = \frac{C^2 \ln(8\sqrt{e}t/\delta)}{2t}. \tag{A.6}$$

Summing up equation (A.6) and (A.5), we have

$$\|f_{t+1} - f^*\|_2^2 \leqslant \frac{C^2 \ln(8\sqrt{e}t/\delta) + 2\kappa Q_2^2 \ln(2t/\delta)\ln^2(t)}{t}$$

as desired. ■

From the bound on L_2 distance, we can immediately get the generalization bound.

Theorem 13 (Generalization bound) *Let the true risk be $R_{true}(f) = \mathbb{E}_{(x,y)}[\ell(f(x), y)]$. Then with probability at least $1-3\delta$ over $(\mathcal{D}^t, \boldsymbol{\omega}^t)$, and C and Q_2 defined as previously*

$$R_{true}(f_{t+1}) - R_{true}(f^*) \leqslant \frac{(C\sqrt{\ln(8\sqrt{e}t/\delta)} + \sqrt{2\kappa}Q_2\sqrt{\ln(2t/\delta)\ln(t)})L}{\sqrt{t}}.$$

Proof By the Lipschitz continuity of $\ell(\cdot, y)$ and Jensen's Inequality, we have

$$R_{true}(f_{t+1}) - R_{true}(f^*) \leqslant L\mathbb{E}_x|f_{t+1}(x) - f^*(x)| \leqslant L\sqrt{\mathbb{E}_x|f_{t+1}(x) - f^*(x)|^2} = L\|f_{t+1} - f^*\|_2.$$

Then the theorem follows from Corollary 36. ■

A.3 Suboptimality

For comprehensive purposes, we also provide the $O(1/t)$ bound for suboptimality.

Corollary 37 *If we set $\gamma_t = \frac{\theta}{t}$ with $\theta\nu = 1$, then the average solution $\hat{f}_{t+1} := \frac{1}{t}\sum_{i=1}^{t} f_i$ satisfies*

$$R(\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[\hat{f}_{t+1}]) - R(f^*) \leqslant \frac{Q(\ln(t)+1)}{t}.$$

where $Q = (4\kappa M^2 + 2\sqrt{2}\kappa^{1/2}LM(\kappa+\phi)Q_1)/\nu$, with Q_1 defined as in Lemma 33.

Proof From the anallysis in Lemma 33,we have

$$\langle h_t - f^*, \bar{g}_t\rangle_{\mathcal{H}} = \frac{1}{2\gamma_t}A_t - \frac{1}{2\gamma_t}A_{t+1} + \gamma_t\mathcal{M}_t + \mathcal{N}_t + \mathcal{R}_t$$

Invoking strongly convexity of $R(f)$, we have $\langle h_t - f^*, \bar{g}_t\rangle \geqslant R(h_t) - R(f^*) + \frac{\nu}{2}\|h_t - f^*\|_{\mathcal{H}}^2$. Taking expectaion on both size and use the bounds in last lemma, we have

$$\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[R(h_t) - R(f^*)] \leqslant (\frac{1}{2\gamma_t} - \frac{\nu}{2})e_t - \frac{1}{2\gamma_t}e_{t+1} + \gamma_t\kappa M^2(1+\nu c_t)^2 + \kappa^{1/2}LB_{1,t}\sqrt{e_t}$$

Assume $\gamma_t = \frac{\theta}{t}$ with $\theta = \frac{1}{\nu}$, then cumulating the above inequalities leads to

$$\sum_{i=1}^{t}\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[R(h_i) - R(f^*)] \leqslant \sum_{i=1}^{t}\gamma_i\kappa M^2(1+\nu c_i)^2 + \sum_{i=1}^{t}\kappa^{1/2}LB_{1,i}\sqrt{e_i}$$

which can be further bounded by

$$\begin{aligned}
\sum_{i=1}^{t}\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[R(h_i) - R(f^*)] &\leqslant \sum_{i=1}^{t}\gamma_i\kappa M^2(1+\nu c_i)^2 + \sum_{i=1}^{t}\kappa^{1/2}LB_{1,i}\sqrt{e_i} \\
&\leqslant \frac{4\kappa M^2}{\nu}\sum_{i=1}^{t}\frac{1}{i} + \frac{2\sqrt{2}\kappa^{1/2}LM(\kappa+\phi)}{\nu}\sum_{i=1}^{t}\sqrt{\frac{e_i}{i}} \\
&\leqslant \frac{4\kappa M^2}{\nu}(\ln(t)+1) + \frac{2\sqrt{2}\kappa^{1/2}LM(\kappa+\phi)}{\nu}Q_1(\ln(t)+1) \\
&= \frac{Q(\ln(t)+1)}{t}
\end{aligned}$$

By convexity, we have $\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[R(\hat{h}_{t+1}) - R(f^*)] \leqslant \frac{Q(\ln(t)+1)}{t}$. The corollary then follows from the fact that $\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[\hat{f}_{t+1}] = \mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[\hat{h}_{t+1}]$ and $R(\mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[\hat{h}_{t+1}]) \leqslant \mathbb{E}_{\mathcal{D}^t,\boldsymbol{\omega}^t}[R(\hat{h}_{t+1})]$. ■

A.3.1 Technical lemma for recursion bounds

Lemma 38 *Suppose the sequence* $\{\Gamma_t\}_{t=1}^{\infty}$ *satisfies* $\Gamma_1 \geqslant 0$, *and* $\forall t \geqslant 1$

$$\Gamma_{t+1} \leqslant \left(1 - \frac{\eta}{t}\right)\Gamma_t + \frac{\beta_1}{t\sqrt{t}}\sqrt{\Gamma_t} + \frac{\beta_2}{t^2},$$

where $\eta > 1, \beta_1, \beta_2 > 0$. *Then* $\forall t \geqslant 1$,

$$\Gamma_t \leqslant \frac{R}{t}, \text{ where } R = \max\left\{\Gamma_1, R_0^2\right\}, R_0 = \frac{\beta_1 + \sqrt{\beta_1^2 + 4(\eta-1)\beta_2}}{2(\eta-1)}.$$

Proof The proof follows by induction. When $t = 1$, it always holds true by the definition of R. Assume the conclusion holds true for t with $t \geqslant 1$, i.e., $\Gamma_t \leqslant \frac{R}{t}$, then we have

$$\begin{aligned}
\Gamma_{t+1} &\leqslant \left(1 - \frac{\eta}{t}\right)\Gamma_t + \frac{\beta_1}{t\sqrt{t}}\sqrt{\Gamma_t} + \frac{\beta_2}{t^2} \\
&= \frac{R}{t} - \frac{\eta R - \beta_1\sqrt{R} - \beta_2}{t^2} \leqslant \frac{R}{t+1} + \frac{R}{t(t+1)} - \frac{\eta R - \beta_1\sqrt{R} - \beta_2}{t^2} \\
&\leqslant \frac{R}{t+1} - \frac{1}{t^2}\left[-R + \eta R - \beta_1\sqrt{R} - \beta_2\right] \leqslant \frac{R}{t+1}
\end{aligned}$$

where the last step can be verified as follows.

$$\begin{aligned}
(\eta-1)R - \beta_1\sqrt{R} - \beta_2 &= (\eta-1)\left[\sqrt{R} - \frac{\beta_1}{2(\eta-1)}\right]^2 - \frac{\beta_1^2}{4(\eta-1)} - \beta_2 \\
&\geqslant (\eta-1)\left[R_0 - \frac{\beta_1}{2(\eta-1)}\right]^2 - \frac{\beta_1^2}{4(\eta-1)} - \beta_2 \geqslant 0
\end{aligned}$$

where the last step follows from the defintion of R_0. ■

Lemma 39 *Suppose the sequence* $\{\Gamma_t\}_{t=1}^{\infty}$ *satisfies*

$$\Gamma_{t+1} \leqslant \frac{\beta_1}{t} + \beta_2\sqrt{\ln(2t/\delta)} \cdot \sum_{i=1}^{t} \frac{\sqrt{\Gamma_i}}{t\sqrt{i}} + \beta_3\sqrt{\ln(\ln(t)/\delta)}\frac{\sqrt{\sum_{i=1}^{t}\Gamma_i}}{t} + \beta_4 \ln(\ln(t/\delta))\frac{1}{t}$$

where $\beta_1, \beta_2, \beta_3, \beta_4 > 0$ *and* $\delta \in (0, 1/e)$. *Then* $\forall 1 \leqslant j \leqslant t (t \geqslant 4)$,

$$\Gamma_j \leqslant \frac{R \ln(2t/\delta) \ln^2(t)}{j},$$

where

$$R = \max\{\Gamma_1, R_0^2\}, R_0 = 2\beta_2 + 2\sqrt{2}\beta_3 + \sqrt{(2\beta_2 + 2\sqrt{2}\beta_3)^2 + \beta_1 + \beta_4}.$$

Proof The proof follows by induction. When $j = 1$ it is trivial. Let us assume it holds true for $1 \leqslant j \leqslant t-1$, therefore,

$$\begin{aligned}
\Gamma_{j+1} &\leqslant \frac{\beta_1}{j} + \beta_2 \sqrt{\ln(2j/\delta)} \cdot \sum_{i=1}^{j} \frac{\sqrt{\Gamma_i}}{j\sqrt{i}} + \beta_3 \sqrt{\ln(\ln(j)/\delta)} \frac{\sqrt{\sum_{i=1}^{j} \Gamma_i}}{j} + \beta_4 \ln(\ln(j/\delta)) \frac{1}{j} \\
&\leqslant \frac{\beta_1}{j} + \beta_2 \sqrt{\ln(2j/\delta)}/j \cdot \sum_{i=1}^{j} \frac{\sqrt{R \ln(2t/\delta) \ln^2(t)}}{i} \\
&\quad + \beta_3 \sqrt{\ln(\ln(j)/\delta)} \frac{\sqrt{\sum_{i=1}^{j} R \ln(2t/\delta) \ln^2(t)/i}}{j} + \beta_4 \ln(\ln(j/\delta)) \frac{1}{j} \\
&\leqslant \frac{\beta_1}{j} + \beta_2 \sqrt{\ln(2j/\delta)}/j \sqrt{R \ln(2t/\delta) \ln^2(t)}(1 + \ln(j)) \\
&\quad + \beta_3 \sqrt{\ln(\ln(j)/\delta)}/j \sqrt{R \ln(2t/\delta) \ln^2(t)} \sqrt{\ln(j) + 1} + \beta_4 \ln(\ln(j/\delta)) \frac{1}{j} \\
&\leqslant \frac{\beta_1}{j} + 2\beta_2 \sqrt{R} \ln(2t/\delta) \ln^2(t)/j + \sqrt{2}\beta_3 \sqrt{R} \ln(2t/\delta) \ln^2(t)/j + \beta_4 \ln(2t/\delta) \frac{1}{j} \\
&\leqslant (2\beta_2 + \sqrt{2}\beta_3)\sqrt{R} \frac{\ln(2t/\delta) \ln^2(t)}{j} + (\beta_1 + \beta_4 \ln(2t/\delta)) \frac{1}{j} \\
&\leqslant \frac{\ln(2t/\delta) \ln^2(t)}{j} [(2\beta_2 + \sqrt{2}\beta_3)\sqrt{R} + \frac{\beta_1}{2} + \frac{\beta_4}{2})
\end{aligned}$$

Since $\sqrt{R} \geqslant 2\beta_2 + 2\sqrt{2}\beta_3 + \sqrt{(2\beta_2 + 2\sqrt{2}\beta_3)^2 + \beta_1 + \beta_4}$, we have $(2\beta_2 + 2\sqrt{2}\beta_3)\sqrt{R} + \frac{\beta_1}{2} + \frac{\beta_4}{2} \leqslant R/2$. Hence, $\Gamma_{j+1} \leqslant \frac{R \ln(2t/\delta) \ln^2(t)}{j+1}$. ■

A.4 Doubly Stochastic Gradient Algorithm for Posterior Variance Operator in Gaussian Process Regression

As we show in Section 3.3.1, the estimation of the variance of the predictive distribution of Gaussian process for regression problem could be recast as estimating the operator Σ defined in (3.10). We first demonstrate that the operator Σ is the solution to the following optimization problem

$$\min_{\Sigma} R(\Sigma) = \frac{1}{2n}\sum_{i=1}^{n} \|k(x_i,\cdot) - \Sigma k(x_i,\cdot)\|_{\mathcal{H}}^2 + \frac{\sigma^2}{2n}\|\Sigma\|_{HS}^2$$

where $\|\cdot\|_{HS}$ is the Hilbert-Schmidt norm of the operator. The gradient of $R(\Sigma)$ with respect to Σ is

$$\nabla R(\Sigma) = \frac{1}{n}\sum_{i=1}^{n}\Big((\Sigma k(x,\cdot) - k(x,\cdot))\otimes k(x,\cdot)\Big) + \frac{\sigma^2}{n}\Sigma = \Sigma\Big(\mathcal{C} + \frac{\sigma^2}{n}I\Big) - \mathcal{C}$$

Set $\nabla R(\Sigma) = 0$, we could obtain the optimal solution, $\mathcal{C}\big(\mathcal{C} + \frac{\sigma^2}{n}I\big)^{-1}$, exactly the same as (3.10).

To derive the doubly stochastic gradient update for Σ, we start with stochastic functional gradient of $R(\Sigma)$. Given $x_i \sim p(x)$, the stochastic functional gradient of $R(\Sigma)$ is

$$\psi(\cdot,\cdot) = \Sigma\Big(\widehat{\mathcal{C}} + \frac{\sigma^2}{n}I\Big) - \widehat{\mathcal{C}}$$

where $\widehat{\mathcal{C}} = k(x_i,\cdot)\otimes k(x_i,\cdot)$ which leads to update

$$\Sigma_{t+1} = \Sigma_t - \gamma_t\psi = \Big(1 - \frac{\sigma^2}{n}\gamma_t\Big)\Sigma_t - \gamma_t\Big(\Sigma_t\widehat{\mathcal{C}}_t - \widehat{\mathcal{C}}_t\Big). \tag{A.7}$$

With such update rule, we could show that $\Sigma_{t+1} = \sum_{i=1,j\geqslant i}^{t}\beta_{ij}^{t+1}k(x_i,\cdot)\otimes k(x_j,\cdot)$ by induction. Let $\Sigma_1 = 0$, then, $\Sigma_2 = \gamma_1 k(x_1,\cdot)\otimes k(x_1,\cdot)$. Assume at t-th iteration, $\Sigma_t = \sum_{i=1,j\geqslant i}^{t-1}\beta_{ij}^{t}k(x_i,\cdot)\otimes k(x_j,\cdot)$, and notice that

$$\Sigma_t\widehat{\mathcal{C}}_t = \Sigma_t^{\top}(\cdot,x_t)\otimes k(x_t,\cdot) = \sum_{i=1}^{t-1}\Big(\sum_{j\geqslant i}^{t-1}\beta_{ij}^{t}k(x_j,x_t)\Big)k(x_i,\cdot)\otimes k(x_t,\cdot),$$

we have $\Sigma_{t+1} = \sum_{i=1,j\geqslant i}^{t} \beta_{ij}^{t+1} k(x_i,\cdot)\otimes k(x_j,\cdot)$ where

$$\begin{aligned}
\beta_{ij}^{t+1} &= \left(1-\frac{\sigma^2}{n}\gamma_t\right)\beta_{ij}^{t}, \quad \forall i \leqslant j < t \\
\beta_{it}^{t+1} &= -\gamma_t \sum_{j=1}^{t} \beta_{ij}^{t} k(x_j, x_t), \quad \forall i < t \\
\beta_{tt}^{t+1} &= \gamma_t
\end{aligned}$$

Recall

$$\widehat{\mathcal{C}}_t = \mathbb{E}_{\omega}[\phi_\omega(x_t)\phi_\omega(\cdot)] \otimes \mathbb{E}_{\omega'}[\phi_{\omega'}(x_t)\phi_{\omega'}(\cdot)] = \mathbb{E}_{\omega,\omega'}[\phi_\omega(x_t)\phi_{\omega'}(x_t)\phi_\omega(\cdot)\otimes\phi_{\omega'}(\cdot)],$$

where ω, ω' are independently sampled from $\rho(\omega)$, we could approximate the $\widehat{\mathcal{C}}_t$ with random features, $\widehat{\mathcal{C}}_t^{\omega,\omega'} = \phi_{\omega_t}(x_t)\phi_{\omega_t'}(x_t)\phi_{\omega_t}(\cdot)\otimes\phi_{\omega_t'}(\cdot)$. Plug random feature approximation into (A.7) leads to

$$\widehat{\Sigma}_{t+1} = \left(1-\frac{\sigma^2}{n}\gamma_t\right)\widehat{\Sigma}_t - \gamma_t\left(\widehat{\Sigma}_t^{\top}(\cdot,x_t)\otimes\phi_{\omega_t'}(x_t)\phi_{\omega_t'}(\cdot) - \widehat{\mathcal{C}}_t^{\omega,\omega'}\right).$$

Therefore, inductively, we could approximate Σ_{t+1} by

$$\widehat{\Sigma}_{t+1} = \sum_{i\leqslant j}^{t} \theta_{ij}^{t}\phi_{\omega_i}(\cdot)\otimes\phi_{\omega_j'}(\cdot)$$

$$\begin{aligned}
\theta_{ij} &= \left(1-\frac{\sigma^2}{n}\gamma_t\right)\theta_{ij}, \, \forall i \leqslant j < t \\
\theta_{it} &= -\gamma_t \sum_{j\geqslant i}^{t-1} \theta_{ij}\phi_{\omega_j'}(x_t)\phi_{\omega_t'}(x_t), \, \forall i < t \\
\theta_{tt} &= \gamma_t\phi_{\omega_t}(x_t)\phi_{\omega_t'}(x_t).
\end{aligned}$$

APPENDIX B
PROOF DETAILS AND EXTENSION OF PARTICLE MIRROR DESCENT IN CHAPTER 4

B.1 Strong convexity

As we discussed, the posterior from Bayes's rule could be viewed as the optimal of an optimization problem in Eq (4.2). We will show that the objective function is strongly convex w.r.t KL-divergence.

Proof for Lemma B.1 The lemma directly results from the generalized Pythagaras theorem for Bregman divergence. Particularly, for KL-divergence, we have

$$KL(q_1||q) \quad = \quad KL(q_1||q_2) + KL(q_2||q) - \langle q_1 - q_2, \nabla\phi(q) - \nabla\phi(q_2)\rangle_2$$

where $\phi(q)$ is the entropy of q.

Notice that $L(q) = KL(q||q^*) - \log Z$, where $q^* = \frac{p(\theta)\Pi_i^N p(x_i|\theta)}{Z}, \quad Z = \int p(\theta)\Pi_i^N p(x_i|\theta)$, we have

$$\begin{aligned}
& KL(q_1||q^*) - KL(q_2||q^*) - \langle q_1 - q_2, \nabla\phi(q_2) - \nabla\phi(q^*)\rangle_2 = KL(q_1||q_2) \\
\Rightarrow \quad & KL(q_1||q^*) - KL(q_2||q^*) - \langle q_1 - q_2, \log q_2 - \log q^*\rangle_2 = KL(q_1||q_2) \\
\Rightarrow \quad & KL(q_1||q^*) - KL(q_2||q^*) - \langle q_1 - q_2, \log q_2 - \log\left(p(\theta)\Pi_i^N p(x_i|\theta)\right)\rangle_2 + \underbrace{\langle q_1 - q_2, \log Z\rangle_2}_{0} \\
& = KL(q_1||q_2) \\
\Rightarrow \quad & L(q_1) - L(q_2) - \langle q_1 - q_2, \nabla L(q_2)\rangle_2 = KL(q_1||q_2)
\end{aligned}$$

■

B.2 Finite Convergence of Stochastic Mirror Descent with Inexact Prox-Mapping in Density Space

Since the prox-mapping of stochastic mirror descent is intractable when directly being applied to the optimization problem (4.2), we propose the *ϵ-inexact prox-mapping* within the stochastic mirror descent framework in Section 4.3. Instead of solving the prox-mapping exactly, we approximate the solution with ϵ error. In this section, we will show as long as the approximation error is tolerate, the stochastic mirror descent algorithm still converges.

Theorem 15 *Denote $q^* = \operatorname{argmin}_{q\in\mathcal{P}} L(q)$, the stochastic mirror descent with inexact prox-mapping after T steps gives*

(a) the recurrence: $\forall t \leqslant T$, $\mathbb{E}[KL(q^*||\tilde{q}_{t+1})] \leqslant \epsilon_t + (1-\gamma_t)\mathbb{E}[KL(q^*||\tilde{q}_t)] + \frac{\gamma_t^2 \mathbb{E}\|g_t\|_\infty^2}{2}$

(b) the sub-optimality: $\mathbb{E}[KL(\bar{q}_T||q^*)] \leqslant \mathbb{E}[L(\bar{q}_T) - L(q^*)] \leqslant \frac{\mathcal{M}^2 \cdot \frac{1}{2}\sum_{t=1}^T \gamma_t^2 + \sum_{t=1}^T \epsilon_t + D_1}{\sum_{t=1}^T \gamma_t}$

where $\bar{q}_T = \sum_{t=1}^T \gamma_t \tilde{q}_t / \sum_{t=1}^T \gamma_t$ and $D_1 = KL(q^*||\tilde{q}_1)$ and $\mathcal{M}^2 := \max_{1\leqslant t\leqslant T} \mathbb{E}\|g_t\|_\infty^2$.

Remark: Based on [70], one can immediately see that, to guarantee the usual rate of convergence, the error ϵ_t can be of order $O(\gamma_t^2)$. The first recurrence implies an overall $O(1/T)$ rate of convergence for the KL-divergence when the stepsize γ_t is as small as $O(1/t)$ and error ϵ_t is as small as $O(1/t^2)$. The second result implies an overall $O(1/\sqrt{T})$ rate of convergence for objective function when larger stepsize $\gamma_t = O(1/\sqrt{T})$ and larger error $\epsilon_t = O(1/t)$ are adopted.

Proof (a) Due to the fact that $\tilde{q}_{t+1}(\theta) \in \mathbf{P}_{\tilde{q}_t}^{\epsilon_t}(\gamma_t g_t)$, *i.e.*,

$$\langle \gamma_t g_t + \log(\tilde{q}_{t+1}) - \log(\tilde{q}_t), \tilde{q}_{t+1} - q\rangle_{L_2} \leqslant \epsilon_t, \forall q \in \mathcal{P},$$

we have

$$\begin{aligned}\langle \gamma_t g_t, \tilde{q}_{t+1} - q\rangle_2 &\leqslant \langle \log(\tilde{q}_t) - \log(\tilde{q}_{t+1}), \tilde{q}_{t+1} - q\rangle_2 + \epsilon_t \\ &= KL(q||\tilde{q}_t) - KL(q||\tilde{q}_{t+1}) - KL(\tilde{q}_{t+1}||\tilde{q}_t) + \epsilon_t\end{aligned}$$

Hence,

$$\langle \gamma_t g_t, \tilde{q}_t - q\rangle_2 \leqslant KL(q||\tilde{q}_t) - KL(q||\tilde{q}_{t+1}) - KL(\tilde{q}_{t+1}||\tilde{q}_t) + \langle \gamma_t g_t, \tilde{q}_t - \tilde{q}_{t+1}\rangle_2 + \epsilon_t. \quad \text{(B.1)}$$

By Young's inequality, we have

$$\langle \gamma_t g_t, \tilde{q}_t - \tilde{q}_{t+1} \rangle_2 \leqslant \frac{1}{2}\|\tilde{q}_t - \tilde{q}_{t+1}\|_1^2 + \frac{\gamma_t^2}{2}\|g_t\|_\infty^2. \tag{B.2}$$

Also, from Pinsker's inequality, we have

$$KL(\tilde{q}_{t+1}||\tilde{q}_t) \geqslant \frac{1}{2}\|\tilde{q}_t - \tilde{q}_{t+1}\|_1^2. \tag{B.3}$$

Therefore, combining (B.1), (B.2), and (B.3), we have $\forall q \in \mathcal{P}$

$$\langle \gamma_t g_t, \tilde{q}_t - q \rangle_2 \leqslant \epsilon_t + KL(q||\tilde{q}_t) - KL(q||\tilde{q}_{t+1}) + \frac{\gamma_t^2}{2}\|g_t\|_\infty^2$$

Plugging q^* and taking expectation on both sides, the LHS becomes

$$\mathbb{E}_x\Big[\langle \tilde{q}_t - q^*, \gamma_t g_t \rangle\Big] \;\; = \;\; \mathbb{E}_x\Big[\langle \tilde{q}_t - q^*, \gamma_t \mathbb{E}[g_t] \rangle \Big| x_{[t-1]}\Big] = \mathbb{E}_x\Big[\langle \tilde{q}_t - q^*, \gamma_t \nabla L(\tilde{q}_t) \rangle\Big],$$

Therefore, we have

$$\mathbb{E}_x\Big[\langle \tilde{q}_t - q^*, \gamma_t \nabla L(\tilde{q}_t) \rangle\Big] \leqslant \epsilon_t + \mathbb{E}_x\big[KL(q^*||\tilde{q}_t)\big] - \mathbb{E}_x\big[KL(q^*||\tilde{q}_{t+1})\big] + \frac{\gamma_t^2}{2}\mathbb{E}_x\|g_t\|_\infty^2 \tag{B.4}$$

Because the objective function is 1-strongly convex w.r.t. KL-divergence,

$$\langle q' - q, \nabla L(q') - \nabla L(q) \rangle = KL(q'||q) + KL(q||q'),$$

and the optimality condition, we have

$$\langle \tilde{q}_t - q^*, \nabla L(\tilde{q}_t) \rangle \geqslant KL(q^*||\tilde{q}_t)$$

we obtain the recursion with inexact prox-mapping,

$$\mathbb{E}_x[KL(q^*||\tilde{q}_{t+1})] \leqslant \epsilon_t + (1-\gamma_t)\mathbb{E}_x[KL(q^*||\tilde{q}_t)] + \frac{\gamma_t^2}{2}\mathcal{M}^2$$

(b) Summing over $t = 1, \ldots, T$ of equation (B.4), we get

$$\sum_{t=1}^{T} \mathbb{E}_x[\langle \tilde{q}_t - q^*, \gamma_t \nabla L(\tilde{q}_t) \rangle] \leqslant \sum_{t=1}^{T} \epsilon_t + KL(q^* || \tilde{q}_1) + \sum_{t=1}^{T} \frac{\gamma_t^2}{2} \mathcal{M}^2$$

By convexity and optimality condition, this leads to

$$\begin{aligned} \left(\sum_{t=1}^{T} \gamma_t \right) \mathbb{E}_x[L(\bar{q}_T) - L(q^*)] &\leqslant \mathbb{E}_x \left[\sum_{t=1}^{T} \gamma_t (L(\tilde{q}_t) - L(q^*)) \right] \\ &\leqslant \sum_{t=1}^{T} \epsilon_t + KL(q^* || \tilde{q}_1) + \sum_{t=1}^{T} \frac{\gamma_t^2}{2} \mathcal{M}^2 \end{aligned}$$

Furthermore, combined with the 1-strongly-convexity, it immediately follows that

$$\mathbb{E}_x[KL(\bar{q}_T || q^*)] \leqslant \mathbb{E}_x[L(\bar{q}_T) - L(q^*)] \leqslant \frac{\frac{1}{2} \sum_{t=1}^{T} \gamma_t^2 \mathcal{M}^2 + \sum_{t=1}^{T} \epsilon_t + D_1}{\sum_{t=1}^{T} \gamma_t}.$$

■

B.3 Convergence Analysis for Integral Approximation

In this section, we provide the details of the convergence analysis of the proposed algorithm in terms of integral approximation w.r.t. the true posterior using a good initialization.

Assume that the prior $p(\theta)$ has support Ω cover true posterior distribution $q^*(\theta)$, then, we could represent

$$q^*(\theta) \in \mathcal{F} = \left\{ q(\theta) = \alpha(\theta) p(\theta), \int \alpha(\theta) p(\theta) d\theta = 1, 0 \leqslant \alpha(\theta) \leqslant C \right\}.$$

Therefore, one can show

Lemma 40 *$\forall q \in \mathcal{F}$, let $\{\theta_i\}_{i=1}^m$ is* i.i.d. *sampled from $p(\theta)$, we could construct $\hat{q}(\theta) = \sum_{i=1}^m \frac{\alpha(\theta_i)\delta(\theta_i)}{\sum_i^m \alpha(\theta_i)}$, such that $\forall f(\theta) : \mathbb{R}^d \to \mathbb{R}$ bounded and integrable,*

$$\mathbb{E}\left[\left| \int \hat{q}(\theta) f(\theta) d\theta - \int q(\theta) f(\theta) d\theta \right| \right] \leqslant \frac{2\sqrt{C} \|f\|_\infty}{\sqrt{m}}.$$

Proof

Given $q(\theta)$, we sample $i.i.d.\{\theta_i\}_{i=1}^m$ from $p(\theta)$, and construct a function

$$\hat{q}(\theta) = \frac{1}{m}\sum_{i=1}^{m}\alpha(\theta_i)\delta(\theta_i,\theta).$$

It is obviously that

$$\mathbb{E}_\theta[\hat{q}(\theta)] = \mathbb{E}_\theta\left[\frac{1}{m}\sum_{i=1}^{m}\alpha(\theta_i)\delta(\theta_i,\theta)\right] = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_\theta\left[\alpha(\theta_i)\delta(\theta_i,\theta)\right] = q(\theta)$$

and

$$\mathbb{E}_\theta\left[\int \hat{q}(\theta)f(\theta)d\theta\right] = \mathbb{E}_\theta\left[\frac{1}{m}\sum_{i=1}^{m}\alpha(\theta_i)f(\theta_i)\right] = \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_\theta\left[\alpha(\theta_i)f(\theta_i)\right] = \int q(\theta)f(\theta)d\theta$$

Then,

$$\begin{aligned}
\mathbb{E}_\theta\left[\left|\int \hat{q}(\theta)f(\theta)d\theta - \int q(\theta)f(\theta)d\theta\right|^2\right] &= \mathbb{E}_\theta\left[\left|\int \hat{q}(\theta)f(\theta)d\theta - \mathbb{E}_\theta\left[\int \hat{q}(\theta)f(\theta)d\theta\right]\right|^2\right] \\
&= \frac{1}{m}\left(\mathbb{E}_\theta\|\alpha(\theta_i)f(\theta_i)\|_2^2 - \|\mathbb{E}_\theta[\alpha(\theta_i)f(\theta_i)]\|_2^2\right) \\
&\leqslant \frac{1}{m}\mathbb{E}_\theta\|\alpha(\theta_i)f(\theta_i)\|_2^2 = \frac{1}{m}\int \alpha(\theta)^2 f(\theta)^2\pi(\theta)d\theta \\
&= \frac{1}{m}\int \alpha(\theta)f(\theta)^2 q(\theta)d\theta \\
&\leqslant \frac{C}{m}\|f(\theta)\|_\infty^2\int \alpha(\theta)q(\theta)d\theta \leqslant \frac{C}{m}\|f(\theta)\|_\infty^2\|\alpha(\theta)\|_\infty
\end{aligned}$$

By Jensen's inequality, we have

$$\begin{aligned}
&\mathbb{E}_\theta\left[\left|\int \hat{q}(\theta)f(\theta)d\theta - \int q(\theta)f(\theta)d\theta\right|\right] \\
&\leqslant \sqrt{\mathbb{E}_\theta\left[\left|\int \hat{q}(\theta)f(\theta)d\theta - \int q(\theta)f(\theta)d\theta\right|^2\right]} \leqslant \frac{\sqrt{C}\|f(\theta)\|_\infty}{\sqrt{m}}
\end{aligned}$$

Apply the above conclusion to $f(\theta) = 1$, we have

$$\mathbb{E}\left[\left|\frac{1}{m}\sum_{i}^{m}\alpha_i - 1\right|\right] \leqslant \frac{\sqrt{C}}{\sqrt{m}}$$

Let $\tilde{q}(\theta) = \frac{\sum_i^m \alpha(\theta_i)\delta(\theta_i,\cdot)}{\sum_i^m \alpha(\theta_i)}$, then $\sum_i^m \frac{\alpha_i}{\sum_i^m \alpha_i} = 1$, and

$$\begin{aligned}
& \mathbb{E}_\theta\left[\left|\int \tilde{q}(\theta)f(\theta)d\theta - \int \hat{q}(\theta)f(\theta)d\theta\right|\right] \\
= \; & \mathbb{E}_\theta\left[\left|\frac{1}{\sum_i^m \alpha(\theta_i)}\sum_i^m \alpha(\theta_i)f(\theta_i) - \frac{1}{m}\sum_i^m \alpha(\theta_i)f(\theta_i)\right|\right] \\
= \; & \mathbb{E}_\theta\left[\left|1 - \frac{\sum_i^m \alpha(\theta_i)}{m}\right|\left\|\frac{1}{\sum_i^m \alpha(\theta_i)}\sum_i^m \alpha(\theta_i)f(\theta_i)\right\|\right] \\
= \; & \mathbb{E}_\theta\left[\left|1 - \frac{\sum_i^m \alpha(\theta_i)}{m}\right|\frac{1}{\sum_i^m \alpha(\theta_i)}\sum_i^m \alpha(\theta_i)|f(\theta_i)|\right] \\
\leqslant \; & \mathbb{E}\left[\left|1 - \frac{\sum_i^m \alpha_i}{m}\|f(\theta)\|_\infty\right|\right] \leqslant \frac{\sqrt{C}\|f(\theta)\|_\infty}{\sqrt{m}}
\end{aligned}$$

Then, we have achieve our conclusion that

$$\begin{aligned}
& \mathbb{E}_\theta\left[\left|\int \tilde{q}(\theta)f(\theta)d\theta - \int q(\theta)f(\theta)d\theta\right|\right] \\
\leqslant \; & \mathbb{E}_\theta\left[\left|\int \hat{q}(\theta)f(\theta)d\theta - \int q(\theta)f(\theta)d\theta\right|\right] + \mathbb{E}_\theta\left[\left|\int \tilde{q}(\theta)f(\theta)d\theta - \int \hat{q}(\theta)f(\theta)d\theta\right|\right] \\
\leqslant \; & \frac{2\sqrt{C}\|f(\theta)\|_\infty}{\sqrt{m}}
\end{aligned}$$

■

With the knowledge of $p(\theta)$ and $q(\theta)$, we set $q_t(\theta) = \alpha_t(\theta)p(\theta)$, the PMD algorithm will reduce to adjust $\alpha(\theta_i)$ for samples $\{\theta_i\}_{i=1}^m \sim \pi(\theta)$ according to the stochastic gradient. Plug the gradient formula into the exact update rule, we have

$$q_{t+1}(\theta) = \frac{q_t(\theta)\exp(-\gamma_t g_t(\theta))}{Z} = \frac{\alpha_t(\theta)\exp(-\gamma_t g_t(\theta))p(\theta)}{Z} = \alpha_{t+1}(\theta)p(\theta)$$

where $\alpha_{t+1}(\theta) = \frac{\alpha_t(\theta)\exp(-\gamma_t g_t(\theta))}{Z}$. Since Z is constant, ignoring it will not effect the multiplicative update.

Given the fact that the objective function, $L(q)$, is 1-*strongly convex* w.r.t. the KL-divergence, we can immediately arrive at the following convergence results as appeared in [70], if we are able to compute the prox-mapping in Eq.(4.3) exactly.

Lemma 41 *One prox-mapping step Eq.(4.3) reduces the error by*

$$\mathbb{E}[KL(q^*||q_{t+1})] \leqslant (1-\gamma_t)\mathbb{E}[KL(q^*||q_t)] + \frac{\gamma_t^2 \mathbb{E}\|g_t\|_\infty^2}{2}.$$

With stepsize $\gamma_t = \frac{\eta}{t}$, it implies

$$\mathbb{E}[KL(q^*||q_T)] \leqslant \max\left\{KL(q^*||q_1), \frac{\eta^2 \mathbb{E}\|g\|_\infty^2}{2\eta - 1}\right\}\frac{1}{T}$$

Proof We could obtain the recursion directly from Theorem 15 by setting $\epsilon = 0$, which means solving the prox-mapping exactly, and the rate of convergence rate could be obtained by solving the recursion as stated in [70]. ■

Lemma 42 *Let q_t is the exact solution of the prox-mapping at t-step, then $\forall f(\theta) : \mathbb{R}^d \to \mathbb{R}$, which is bounded and integrable, we have*

$$\mathbb{E}\left[\left|\int q_t(\theta)f(\theta)d\theta - \int q(\theta)f(\theta)d\theta\right|\right] \leqslant \max\left\{\sqrt{KL(q^*||q_1)}, \frac{\eta\mathbb{E}\|g\|_\infty}{\sqrt{2\eta-1}}\right\}\frac{\|f\|_\infty}{\sqrt{t}}.$$

Proof

$$\begin{aligned}
&\mathbb{E}\left[\left|\int q_t(\theta)f(\theta)d\theta - \int q^*(\theta)f(\theta)d\theta\right|\right] = \mathbb{E}\|\langle q_t(\theta) - q^*(\theta)\,, f(\theta)\rangle_{L_2}\|_2 \\
&\leqslant \mathbb{E}[\|q_t(\theta) - q^*(\theta)\|_1\|f\|_\infty] \leqslant \|f\|_\infty \mathbb{E}[\|q_t(\theta) - q^*(\theta)\|_1] \leqslant \|f\|_\infty \mathbb{E}\left[\sqrt{\frac{1}{2}KL(q^*||q_t)}\right] \\
&\leqslant \max\left\{\sqrt{KL(q^*||q_1)}, \frac{\eta\mathbb{E}\|g\|_\infty}{\sqrt{2\eta-1}}\right\}\frac{\|f\|_\infty}{\sqrt{t}}
\end{aligned}$$

The second last inequality comes from Pinsker's inequality. ■

Theorem 18 *Assume the particle proposal prior $p(\theta)$ has the same support as the true posterior $q^*(\theta)$,* i.e., *$0 \leqslant q^*(\theta)/p(\theta) \leqslant C$. With further condition about the model $\|p(x|\theta)^N\|_\infty \leqslant \rho, \forall x$, then $\forall f(\theta) : \mathbb{R}^d \to \mathbb{R}$ bounded and integrable, with stepsize $\gamma_t = \frac{\eta}{t}$, the PMD algorithm*

return m weighted particles after T iteration such that

$$\begin{aligned}&\mathbb{E}\left[\left|\int \tilde{q}_t(\theta)f(\theta)d\theta - \int q^*(\theta)f(\theta)d\theta\right|\right]\\ \leqslant\ & \frac{2\sqrt{\max\{C, \rho\exp(\|g(\theta)\|_\infty)\}}\|f\|_\infty}{\sqrt{m}} + \max\left\{\sqrt{KL(q^*||\pi)}, \frac{\eta\mathbb{E}\|g\|_\infty}{\sqrt{2\eta-1}}\right\}\frac{\|f\|_\infty}{\sqrt{T}}.\end{aligned}$$

Proof We first decompose the error into optimization error and finite approximation error.

$$\begin{aligned}&\mathbb{E}\left[\left|\int \tilde{q}_t(\theta)f(\theta)d\theta - \int q^*(\theta)f(\theta)d\theta\right|\right]\\ \leqslant\ & \underbrace{\mathbb{E}\left[\left|\int \tilde{q}_t(\theta)f(\theta)d\theta - \int q_t(\theta)f(\theta)d\theta\right|\right]}_{\text{finite approximation error } \epsilon_1} + \underbrace{\mathbb{E}\left[\left|\int q_t(\theta)f(\theta)d\theta - \int q^*(\theta)f(\theta)d\theta\right|\right]}_{\text{optimization error } \epsilon_2}\end{aligned}$$

For the optimization error, by lemma 42, we have

$$\epsilon_2 \leqslant \max\left\{\sqrt{KL(q^*||q_1)}, \frac{\eta\mathbb{E}\|g\|_\infty}{\sqrt{2\eta-1}}\right\}\frac{\|f\|_\infty}{\sqrt{t}}.$$

Recall that

$$\begin{aligned}q_t(\theta) &= \frac{q_{t-1}(\theta)\exp(-\gamma_{t-1}g_{t-1}(\theta))}{Z}\\ &= \frac{\alpha_{t-1}(\theta)\pi(\theta)(\alpha_{t-1}^{-\gamma_{t-1}}(\theta)p(x|\theta)^{N\gamma_{t-1}})}{Z} = \alpha_{t-1}^{1-\gamma_{t-1}}(\theta)\pi(\theta)\frac{p(x|\theta)^{N\gamma_{t-1}}}{Z}\end{aligned}$$

which results the update $\alpha_t(\theta) = \frac{\alpha_{t-1}^{1-\gamma_{t-1}}(\theta)p(x|\theta)^{N\gamma_{t-1}}}{Z}$. Notice $Z = \int q_t(\theta)\exp(-\gamma_t g_t(\theta))d\theta$, we have $\exp(-\gamma_t\|g_t(\theta)\|_\infty) \leqslant Z \leqslant \exp(\gamma_t\|g_t(\theta)\|_\infty)$. By induction, it can be show that $\|\alpha_t\|_\infty \leqslant \max\{C, \rho\exp(\|g_t(\theta)\|_\infty)\} \leqslant \max\{C, \rho\exp(\|g(\theta)\|_\infty)\}$. Therefore, by lemma 40, we have

$$\epsilon_1 \leqslant \frac{2\sqrt{\max\{C, \rho\exp(\|g(\theta)\|_\infty)\}}\|f\|_\infty}{\sqrt{m}}.$$

Combine ϵ_1 and ϵ_2, we achieve the conclusion. ■

Remark: Simply induction without the assumption from the update of $\alpha_t(\theta)$ will result the

upper bound of sequence $\|\alpha_t\|_\infty$ growing. The growth of sequence $\|\alpha_t\|_\infty$ is also observed in the proof [113] for sequential Monte Carlo on dynamic models. To achieve the uniform convergence rate for SMC of inference on dynamic system, [113, 122] require the models should satisfy i), $\epsilon\nu(\theta_i) \leqslant p(x_i|\theta_i)p(\theta_i|\theta_{i-1}) \leqslant \epsilon^{-1}\nu(\theta_i)$, $\forall x$ where $\nu(\theta)$ is a positive measure, and ii), $\frac{\sup_\theta p(x|\theta)}{\inf_{\mu\in\mathcal{P}}\langle\mu(\theta)p(\cdot|\theta)p(x|\cdot)\rangle} \leqslant \rho$. Such a rate is only for SMC on dynamic system. For static model, the trandistiion distribution is unknown, and therefore, no guarantee is provided yet. With much simpler and more generalized condition on the model, *i.e.*, $\|p(x|\theta)^N\|_\infty \leqslant \rho$, we also achieve the uniform convergence rate for static model. There are plenties of models satisfying this condition. We list several such models below.

1. logistic regression, $p(y|x,w) = \frac{1}{1+\exp(-yw^\top x)}$, and $\|p(y|x,w)\|_\infty \leqslant 1$.

2. probit regression, $p(y=1|x,w) = \Phi(w^\top x)$ where $\Phi(\cdot)$ is the cumulative distribution function of normal distribution. $\|p(y|x,w)\|_\infty \leqslant 1$.

3. multi-category logistic regression, $p(y=k|x,W) = \frac{\exp(w_k^\top x)}{\sum_{i=1}^K \exp(w_k^\top x)}$, and $\|p(y|x,W)\|_\infty \leqslant 1$.

4. latent Dirichlet allocation,

$$\begin{aligned} p(x_d|\theta_d,\Phi) &= \mathbb{E}_{z_d\sim p(z_d|\theta_d)}[p(x_d|z_d,\Phi)] \\ p(x_d|z_d,\Phi) &= \prod_{n=1}^{N_d}\prod_{w=1}^{W}\prod_{k=1}^{K}\Phi_{kw}^{z_{dnk}x_{dnw}} \\ p(z_d|\theta_d) &= \prod_{n=1}^{N_d}\prod_{k=1}^{K}\theta_{dk}^{z_{dnk}} \end{aligned}$$

   and $\|p(x_d|\theta_d,\Phi)\|_\infty \leqslant \max_{z_d}\|p(x_d|z_d,\Phi)\|_\infty \leqslant 1$.

5. linear regression, $p(y|w,x) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-(y-w^\top x)^2/2\sigma^2)$, and $\|p(y|w,x)\|_\infty \leqslant \frac{1}{\sigma\sqrt{2\pi}}$.

6. Gaussian model and PCA, $p(x|\mu,\Sigma) = (2\pi\det(\Sigma))^{-\frac{1}{d}}\exp\left(-\frac{1}{2}(x-\mu)^\top\Sigma(x-\mu)\right)$, and $\|p(x|\mu,\Sigma)\|_\infty \leqslant (2\pi\det(\Sigma))^{-\frac{1}{d}}$.

B.4 Error Bound of Weighted Kernel Density Estimator

Before we start to prove the finite convergence in general case, we need to characterize the error induced by weighted kernel density estimator. In this section, we analyze the error in terms of both L_1 and L_2 norm, which are used for convergence analysis measured by KL-divergence in Appendix B.5 .

B.4.1 L_1-Error Bound of Weighted Kernel Density Estimator

We approximate the density function $q(\theta)$ using the weighted kernel density estimator $\tilde{q}(\theta)$ and would like to bound the L_1 error, i.e. $\|\tilde{q}(\theta) - q(\theta)\|_1$ both in expectation and with high probability. We consider an unnormalized kernel density estimator as the intermediate quantity

$$\varrho_m(\theta) = \frac{1}{m}\sum_{i=1}^{m} \omega(\theta_i) K_h(\theta, \theta_i)$$

Note that $\mathbb{E}[\varrho_m(\theta)] = \mathbb{E}_{\theta_i}[\omega(\theta_i)K_h(\theta,\theta_i)] = q \star K_h$. Then the error can be decomposed into three terms as

$$\epsilon := \mathbb{E}\,\|\tilde{q}(\theta) - q(\theta)\|_1 \leqslant \underbrace{\mathbb{E}\,\|\tilde{q}(\theta) - \varrho_m(\theta)\|_1}_{\text{normalization error}} + \underbrace{\mathbb{E}\,\|\varrho_m(\theta) - \mathbb{E}\,\varrho_m(\theta)\|_1}_{\text{sampling error (variance)}} + \underbrace{\|\mathbb{E}\,\varrho_m(\theta) - q(\theta)\|_1}_{\text{approximation error (bias)}}$$

We now present the proof for each of these error bounds.

To formally show that, we begin by giving the definition of a special class of kernels and Hölder classes of densities that we consider.

Definition 43 ($(\beta;\mu,\nu,\delta)$-valid density kernel) *We say a kernel function $K(\cdot)$ is a $(\beta;\mu,\nu)$-valid density kernel, if $K(\theta,\theta) = K(\theta-\theta)$ is a bounded, compactly supported kernel such that*

(i) $\int K(z)dz = 1$

(ii) $\int |K(z)|^r dz \leqslant \infty$ *for any* $r \geqslant 1$, *particularly,* $\int K(z)^2\,dz \leqslant \mu^2$ *for some* $\mu > 0$.

(iii) $\int z^s K(z)dz = 0$, *for any* $s = (s_1,\ldots,s_d) \in \mathbb{N}^d$ *such that* $1 \leqslant |s| \leqslant \lfloor\beta\rfloor$. *In addition,* $\int \|z\|^\beta |K(z)|dz \leqslant \nu$ *for some* $\nu > 0$.

For simplicity, we sometimes call $K(\cdot)$ as a β-valid density kernel if the constants μ and ν are not specifically given. Notice that all spherically symmetric compactly supported probability density and product kernels based on compactly supported symmetric univariate densities satisfy the conditions. For instance, the kernel $K(\theta) = (2\pi)^{-d/2}\exp(-\|\theta\|^2/2)$ satisfies the conditions with $\beta = \infty$, and it is used through out our experiments. Furthermore, we will focus on a class of smooth densities

Definition 44 ($(\beta;\mathcal{L})$-Hölder density function) *We say a density function $q(\cdot)$ is a $(\beta;\mathcal{L})$-Hölder density function if function $q(\cdot)$ is $\lfloor\beta\rfloor$-times continuously differentiable on its support Ω and satisfies*

(i) for any z_0, there exists $L(z_0) > 0$ such that

$$|q(z) - q_{z_0}^{(\beta)}(z)| \leqslant L(z_0)\|z - z_0\|^{\beta}, \forall z \in \Omega$$

where $q_{z_0}^{(\beta)}$ is the $\lfloor\beta\rfloor$-order Taylor approximation, i.e.

$$q_{z_0}^{(\beta)}(z) := \sum_{s=(s_1,\ldots,s_d):|s|\leqslant\lfloor\beta\rfloor} \frac{(z - z_0)^s}{s!} D^s q(z_0);$$

(ii) in addition, the integral $\int L(z)dz \leqslant \mathcal{L}$.

$f \in C_{\mathcal{L}}^{\beta}(\Omega)$ means f is $(\beta;\mathcal{L})$-Hölder density function.

Then given the above setting for the kernel function and the smooth densities, we can characterize the error of the weighted kernel density estimator as follows.

KDE error due to bias

Lemma 45 (Bias) *If $q(\cdot) \in C_{\mathcal{L}}^{\beta}(\Omega)$ and K is a $(\beta;\mu,\nu)$-valid density kernel, then*

$$\|q(\theta) - \mathbb{E}[\varrho_m(\theta)]\|_1 \leqslant \nu\mathcal{L}h^{\beta}.$$

Proof The proof of this lemma follows directly from Chapter 4.3 in [218].

$$\begin{aligned}
|\mathbb{E}[\varrho_m(\theta)] - q(\theta)| &= |q \star K_h(\theta) - q(\theta)| \\
&= \int \frac{1}{h^d} K(\frac{z-\theta}{h}) q(z) dz - q(\theta) \\
&= \int \frac{1}{h^d} K(\frac{z}{h}) [q(\theta+z) - q(\theta)] dz \\
&= \int K(z)[q(\theta+hz) - q(\theta)] dz \\
&\leqslant \left| \int K(z)[q(\theta+hz) - q_\theta^{(\beta)}(\theta+hz)] dz \right| \\
&\quad + \int \left| K(z)[q_\theta^{(\beta)}(\theta+hz) - q(\theta)] dz \right| \\
&\leqslant L(\theta) \int |K(z)| \|hz\|^\beta dz + \left| \int K(z)[q_\theta^{(\beta)}(\theta+hz) - q(\theta)] dz \right|
\end{aligned}$$

Note that $q_\theta^{(\beta)}(\theta+hz) - q(\theta)$ is a polynomial of degree at most $\lfloor \beta \rfloor$ with no constant, by the definition of $(\beta; \mu, \nu)$-valid density kernel, the second term is zero. Hence, we have $|\mathbb{E}[\varrho_m(\theta)] - q(\theta)| \leqslant \nu L(\theta) h^\beta$, and therefore

$$\|\mathbb{E}[\varrho_m(\theta)] - q(\theta)\|_1 \leqslant \nu h^\beta \int L(\theta) d\theta \leqslant \nu \mathcal{L} h^\beta.$$

■

KDE error due to variance

The variance term can be bounded using similar techniques as in [219].

Lemma 46 (Variance) *Assume $\omega\sqrt{p} \in L_1$ with bounded support, then*

$$\mathbb{E} \left\| \varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)] \right\|_1 \leqslant \frac{\mu}{\sqrt{m} h^{\frac{d}{2}}} \int \omega \sqrt{p} \, d\theta + o((mh^d)^{-\frac{1}{2}}).$$

Proof For any θ, we have

$$\begin{aligned}\sigma^2(\theta) :&= \mathbb{E}\left[(\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)])^2\right] \\ &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}[\omega^2(\theta_i)K_h^2(\theta,\theta_i)] - (q \star K_h)^2 \leqslant \frac{(\omega^2 q)\star K_h^2}{m}\end{aligned}$$

Denote $\mu(K) := \sqrt{\int K(\theta)^2\, d\theta}$ and kernel $K^+(\theta) = \frac{K^2(\theta)}{\mu(K)^2}$, then $\mu(K) \leqslant \mu$, $\int K^+ d\theta = 1$ and

$$K_h^+(\theta) = \frac{1}{h^d}K^+(\theta/d) = \frac{1}{h^d}\frac{K(\theta/h)K(\theta/h)}{\mu^2(K)} = \frac{h^d}{\mu^2(K)}K_h^2(\theta)$$

Hence,

$$\sigma^2(\theta) \leqslant \frac{\mu^2(K)(\omega^2 p)\star K_h^+}{mh^d} \leqslant \frac{\mu^2[(\omega^2 p)\star K_h^+ - \omega^2 p]}{mh^d} + \frac{\mu^2(\omega^2 p)}{mh^d}.$$

Note that $\sigma(\theta) = \sqrt{\mathbb{E}\left[(\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)])^2\right]} \geqslant \mathbb{E}|\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)]|$, hence

$$\begin{aligned}&\mathbb{E}\left\|\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)]\right\|_1 \\ &= \int \mathbb{E}|\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)]|\, d\theta \leqslant \int \sigma(\theta)\, d\theta \\ &\leqslant \int \sqrt{\frac{\mu^2(\omega^2 p)\star K_h^+ - \omega^2 p]}{mh^d}} + \sqrt{\frac{\mu^2(\omega^2 p)}{mh^d}}\, d\theta \\ &\leqslant \frac{\mu}{\sqrt{m}h^{d/2}}\left[\int \sqrt{\omega^2 p}\, d\theta + \int \sqrt{(\omega^2 p)\star K_h^+ - \omega^2 p}\, d\theta\right] \\ &\leqslant \frac{\mu}{\sqrt{m}h^{d/2}}\left[\int \omega\sqrt{p}\, d\theta + \sqrt{|\Omega|}\cdot\sqrt{\int \left|(\omega^2 p)\star K_h^+ - \omega^2 p\right|\, d\theta}\right]\end{aligned}$$

From Theorem 2.1 in [219], we have $\int \left|(\omega^2 p)\star K_h^+ - \omega^2 p\right|\, d\theta = o(1)$. Therefore, we conclude that

$$\mathbb{E}\left\|\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)]\right\|_1 \leqslant \frac{\mu}{\sqrt{m}h^{d/2}}\|\omega\sqrt{p}\|_1 + o((mh^d)^{-\frac{1}{2}}).$$

■

KDE error due to normalization

The normalization error term can be easily derived based on the variance.

Lemma 47 (Normalization error) *Assume* $\omega\sqrt{p} \in L_2$

$$\mathbb{E}\left\|\tilde{q}(\theta) - \varrho_m(\theta)\right\|_1 \leqslant \frac{1}{\sqrt{m}} \left(\int \omega^2(\theta) p(\theta)\, d\theta \right)^{1/2}.$$

Proof Denote $\omega_i := \omega(\theta_i)$, then $\mathbb{E}[\omega_i] = \int \omega(\theta) p(\theta)\, d\theta = 1$ and $\mathbb{E}[\omega_i^2] = \int \omega^2(\theta) p(\theta)\, d\theta$, for any $i = 1, \ldots, m$. Hence,

$$\mathbb{E}|\frac{1}{m}\sum_{i=1}^{m} \omega_i - 1|^2 = \frac{1}{m} \int \omega^2(\theta) p(\theta)\, d\theta.$$

Recall that $\tilde{q}(\theta) = \frac{1}{\sum_{i=1}^m \omega_i} \sum_{i=1}^m \omega_i K_h(\theta, \theta_i)$ and $\varrho_m(\theta) = \frac{1}{m} \sum_{i=1}^m \omega_i K_h(\theta, \theta_i)$.

$$\begin{aligned}
&\mathbb{E}\left\|\tilde{q}(\theta) - \varrho_m(\theta)\right\|_1 \\
&\leqslant \mathbb{E}\left\| \frac{1}{\sum_{i=1}^m \omega_i} \sum_{i=1}^m \omega_i K_h(\theta, \theta_i) - \frac{1}{m} \sum_{i=1}^m \omega_i K_h(\theta, \theta_i) \right\|_1 \\
&\leqslant \mathbb{E}\left\| \left| 1 - \frac{\sum_{i=1}^m \omega_i}{m} \right| \frac{1}{\sum_{i=1}^m \omega_i} \sum_{i=1}^m \omega_i K_h(\theta, \theta_i) \right\|_1 \\
&\leqslant \mathbb{E}\left| 1 - \frac{\sum_{i=1}^m \omega_i}{m} \right| \cdot \|K_h(\theta)\|_1
\end{aligned}$$

Since $\|K_h\|_1 = \int \frac{1}{h^d} K(\theta/h)\, d\theta = \int K(\theta)\, d\theta = 1$, we have

$$\mathbb{E}\left\|\tilde{q}(\theta) - \varrho_m(\theta)\right\|_1 \leqslant \frac{1}{\sqrt{m}} \sqrt{\int \omega^2(\theta) p(\theta)\, d\theta} = \frac{1}{\sqrt{m}} \|w\sqrt{p}\|_2$$

■

KDE error in expectation and with high probability

Based on the above there lemmas, namely, Lemma 45 - 47, we can immediately arrive at the bound of the L_1 error in expectation as stated in Theorem 17. We now provide the proof for the high probability bound as stated below.

Corollary 48 (Overall error in high probability) *Besides the above assumption, let us also assume that $\omega(\theta)$ is bounded, i.e. there exists $0 < B_1 \leqslant B_2 < \infty$ such that $B_1 \leqslant \omega(\theta) \leqslant B_2, \forall\theta$. Then, with probability at least $1-\delta$,*

$$\begin{aligned}\|\tilde{q}(\theta) - q(\theta)\|_1 \;\leqslant\; & \nu\mathcal{L}h^\beta + \frac{\mu}{\sqrt{m}h^{d/2}}\|\omega\sqrt{p}\|_1 + \frac{1}{\sqrt{m}}\|\omega\sqrt{p}\|_2 \\ & + \frac{1}{\sqrt{m}}\sqrt{8B_1B_2\log(1/\delta)} + o((mh^d)^{-\frac{1}{2}}).\end{aligned}$$

Proof We use McDiarmid's inequality to show that the function $f(\Theta) = \|\tilde{q}(\theta) - q(\theta)\|_1$, defined on the random data $\Theta = (\theta_1, \ldots, \theta_m)$, is concentrated on the mean. Let $\tilde{\Theta} = (\theta_1, \ldots, \tilde{\theta}_j, \ldots, \theta_m)$. We denote

$$\omega = (\omega(\theta_1), \ldots, \omega(\theta_m)), \quad \tilde{\omega} = (\omega(\theta_1), \ldots, \omega(\tilde{\theta}_j), \ldots, \omega(\theta_m)).$$

Denote $k = (K_h(\theta, \theta_1), \ldots, K_h(\theta, \theta_m))$ and $\tilde{k} = (K_h(\theta, \theta_1), \ldots, K_h(\theta, \theta'_j), \ldots, K_h(\theta, \theta_m))$. We first show that $|f(\Theta) - f(\Theta')|$ is bounded.

$$\begin{aligned}
& |f(\Theta) - f(\Theta') \\
= \;& \left| \|\tilde{q}_\Theta(\theta) - q(\theta)\|_1 - \left\|\tilde{q}_{\tilde{\Theta}}(\theta) - q(\theta)\right\|_1 \right| \\
\leqslant \;& \left\|\tilde{q}_\Theta(\theta) - \tilde{q}_{\tilde{\Theta}}(\theta)\right\|_1 \\
= \;& \left\| \frac{\sum_{i=1}^m \omega_i k_i}{\sum_{i=1}^m \omega_i} - \frac{\sum_{i=1}^m \tilde{\omega}_i \tilde{k}_i}{\sum_{i=1}^m \tilde{\omega}_i} \right\|_1 \\
\leqslant \;& \left\| \frac{(\tilde{\omega}_j - \omega_j) \cdot (\sum_{i=1}^m \omega_i k_i) - (\sum_{i=1}^m \omega_i)(\tilde{\omega}_i \tilde{k}_i - \omega_j k_j)}{(\sum_{i=1}^m \omega_i) \cdot (\sum_{i=1}^m \tilde{\omega}_i)} \right\|_1 \\
\leqslant \;& \left\| \frac{\tilde{\omega}_j - \omega_j}{(\sum_{i=1}^m \tilde{\omega}_i)} \right\|_\infty + \left\| \frac{\tilde{\omega}_i \tilde{k}_i - \omega_j k_j}{\sum_{i=1}^m \tilde{\omega}_i} \right\|_1 \\
\leqslant \;& \frac{2B_1B_2}{m} + \frac{2B_1B_2}{m} \leqslant \frac{4B_1B_2}{m}
\end{aligned}$$

Invoking the McDiamid's inequality, we have

$$\Pr\left(f(\Theta) - \mathbb{E}_\Theta[f(\Theta)] \geqslant \epsilon\right) \leqslant \exp\left\{-\frac{m\epsilon^2}{8B_1^2B_2^2}\right\}, \forall\epsilon > 0$$

which implies the corollary.

■

B.4.2 L_2-Error Bound of Weighted Kernel Density Estimator

Following same argument yields also similar L_2-error bound of the weighted kernel density estimator, i.e. $\|\tilde{q}(\theta) - q(\theta)\|_2$. For completeness and also for future reference, we provide the exact statement of the bound below in line with Theorem 17 and Corollary 48.

Theorem 49 (L_2-error in expectation) *Let $q = \omega p \in C^{\beta}_{\mathcal{L}}(\Omega)$ and K be a $(\beta; \mu, \nu)$-valid density kernel. Assume that $\omega^2 p \in L_2$ and has bounded support. Then*

$$\mathbb{E}\,\|\tilde{q}(\theta) - q(\theta)\|_2^2 \leqslant 2(\nu h^{\beta}\mathcal{L})^2 + \frac{8\mu^2}{mh^d}\|\omega\sqrt{p}\|_2^2 + o((mh^d)^{-1}).$$

Proof The square L_2-error can also be decomposed into three terms.

$$\mathbb{E}\,\|\tilde{q}(\theta) - q(\theta)\|_2^2 \leqslant 4\underbrace{\mathbb{E}\,\|\tilde{q}(\theta) - \varrho_m(\theta)\|_2}_{\text{normalization error}} + 4\underbrace{\mathbb{E}\,\|\varrho_m(\theta) - \mathbb{E}\,\varrho_m(\theta)\|_2^2}_{\text{sampling error (variance)}} + 2\underbrace{\|\mathbb{E}\,\varrho_m(\theta) - q(\theta)\|_2^2}_{\text{approximation error (bias)}}$$

This uses the inequality $(a+b+c)^2 \leqslant 2a^2 + 4b^2 + 4c^2$ for any a, b, c. From Lemma 45, we already have $|\mathbb{E}[\varrho_m(\theta)] - q(\theta)| \leqslant L(\theta)\int |K(z)\|hz\|^{\beta}dz, \forall\theta$. Hence,

$$\|\mathbb{E}[\varrho_m(\theta)] - q(\theta)\|_2^2 \leqslant \nu^2 h^{2\beta}\int L^2(\theta)d\theta \leqslant (\nu h^{\beta}\mathcal{L})^2. \tag{B.5}$$

From proof for Lemma 46, we have

$$\mathbb{E}\,\|\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)]\|_2^2 = \int \mathbb{E}|\varrho_m(\theta) - \mathbb{E}[\varrho_m(\theta)]|^2\,d\theta \leqslant \int \sigma^2(\theta)\,d\theta \tag{B.6}$$

$$\leqslant \int \tfrac{\mu^2[(\omega^2 p)\star K_h^+ - \omega^2 p]}{mh^d} + \tfrac{\mu^2(\omega^2 p)}{mh^d}\,d\theta \leqslant \tfrac{\mu^2}{mh^d}\|\omega\sqrt{p}\|_2^2 + o((mh^d)^{-1}) \tag{B.7}$$

In addition, we have for the normalization error term,

$$\begin{aligned}\mathbb{E}\left\|\tilde{q}(\theta)-\varrho_m(\theta)\right\|_2^2 &\leqslant \mathbb{E}\left\|\left(1-\frac{\sum_{i=1}^m \omega_i}{m}\right)\frac{\sum_{i=1}^m \omega_i K_h(\theta,\theta_i)}{\sum_{i=1}^m \omega_i}\right\|_2^2 \\ &\leqslant \mathbb{E}\left|1-\frac{\sum_{i=1}^m \omega_i}{m}\right|^2 \cdot \|K_h\|_2^2 \leqslant \frac{\mu^2}{mh^d}\|\omega\sqrt{p}\|_2^2\end{aligned} \tag{B.8}$$

Combining equation (B.5) , (B.6) and (B.8), it follows that

$$\mathbb{E}\left\|\tilde{q}(\theta)-q(\theta)\right\|_2^2 \leqslant 2(\nu h^\beta \mathcal{L})^2 + \frac{8\mu^2}{mh^d}\|\omega\sqrt{p}\|_2^2 + o((mh^d)^{-1}).$$

■

Corollary 50 (L_2-error in high probability) *Besides the above assumption, let us also assume that $\omega(\theta)$ is bounded, i.e. there exists $0 < B_1 \leqslant B_2 < \infty$ such that $B_1 \leqslant \omega(\theta) \leqslant B_2, \forall\theta$. Then, with probability at least $1-\delta$,*

$$\left\|\tilde{q}(\theta)-q(\theta)\right\|_2^2 \leqslant 2(\nu h^\beta \mathcal{L})^2 + \frac{8\mu^2}{mh^d}\|\omega\sqrt{p}\|_2^2 + o((mh^d)^{-1}) + \frac{16B_1B_2\mu^2}{m}\sqrt{\log(1/\delta)}.$$

Proof Use McDiarmid's inequality similar as proof for Corollary 48. ■

B.5 Convergence Analysis for Density Approximation

In this section, we consider the rate of convergence for the entire density measured by KL-divergence. We start with the following lemma that show the renormalization does not effect the optimization in the sense of optimal, and we show the importance weight $\omega_t(\theta) = \frac{\exp(-\gamma_t g_t(\theta))}{Z}$ at each step are bounded under proper assumptions. Moreover, the error of the prox-mapping at each step incurred by the weighted density kernel density estimation is bounded.

Lemma 51 *Let $\zeta = \int_{\setminus\Omega} \tilde{q}_t d\theta$, $\widehat{q}_t = \frac{\tilde{q}_t}{1-\zeta}$ is a valid density on Ω, then, $\tilde{q}_t^+ = \widehat{q}_t^+$, where $\tilde{q}_t^+ := \operatorname{argmin}_{q\in\mathcal{P}(\Omega)} F_t(q;\tilde{q}_t)$, $\widehat{q}_t^+ := \operatorname{argmin}_{q\in\mathcal{P}(\Omega)} F_t(q;\widehat{q}_t)$, and $F_t(q;q') := \langle q, \gamma_t g\rangle_{L_2} + KL(q\|q')$.*

Proof The minima of prox-mapping is not effected by the renormalization. Indeed, such a fact can be verified by comparing to $\tilde{q}_t^+ = \operatorname{argmin} F_t(q; \tilde{q}_t)$ and $\widehat{q}_t^+ = \operatorname{argmin} F_t(q; \widehat{q}_t)$, respectively.

$$\widehat{q}_t^+ = \frac{(\frac{1}{1-\zeta}\tilde{q}_t)^{1-\gamma_t} p(\theta)_t^{\gamma} p(x_t|\theta)^{N\gamma_t}}{\int (\frac{1}{1-\zeta}\tilde{q}_t)^{1-\gamma_t} p(\theta)_t^{\gamma} p(x_t|\theta)^{N\gamma_t} d\theta} = \frac{\tilde{q}_t^{1-\gamma_t} p(\theta)_t^{\gamma} p(x_t|\theta)^{N\gamma_t}}{\int \tilde{q}_t^{1-\gamma_t} p(\theta)_t^{\gamma} p(x_t|\theta)^{N\gamma_t} d\theta} = \tilde{q}_t^+$$

■

Due to the fact, we use $\tilde{q}_t^+$ following for consistency. Although the algorithm updates based on $\tilde{q}_t$, it is implicitly doing renoramlization after each update. We will show that $\widehat{q}_{t+1}$ is an ϵ-inexact prox-mapping.

Lemma 52 *Assume for all mini-batch of examples $\|g_t(\theta)\|_\infty^2 \leqslant M^2$, then we have*

(a) $\exp(-2\gamma_t M) \leqslant \omega_t(\theta) = \frac{\tilde{q}_t^+(\theta)}{\widehat{q}_t(\theta)} \leqslant \exp(2\gamma_t M)$,

(b) $\|\nabla F_t(\tilde{q}_t^+; \widehat{q}_t)\|_\infty \leqslant 3\gamma_t M$.

Proof Let $Z := \int q_t(\theta) \exp(-\gamma_t g_t(\theta)) d\theta$. We have $\exp(-\gamma_t M) \leqslant Z \leqslant \exp(\gamma_t M)$.
(a) Since $\|g_t(\theta)\|_\infty^2 \leqslant M^2$, we have

$$\exp(-2\gamma_t M) \leqslant \omega_t(\theta) = \frac{\tilde{q}_t^+(\theta)}{\widehat{q}_t(\theta)} = \frac{\exp(-\gamma_t g_t(\theta))}{Z} \leqslant \exp(2\gamma_t M).$$

(b) Also, because $\nabla F_t(q_t^+) = \gamma_t g_t + \log \frac{\tilde{q}_t^+}{\widehat{q}_t} = \gamma_t g_t + \log(\omega_t)$, it immediately follows

$$\|\nabla F_t(\tilde{q}_t^+; \widehat{q}_t)\|_\infty \quad = \quad \|\gamma_t g_t + \log(\omega_t)\|_\infty \leqslant \gamma_t \|g_t\|_\infty + \|\log(\omega_t)\|_\infty \leqslant \gamma_t M + (2\gamma_t M) = 3\gamma_t M.$$

■

Lemma 53 *Let $\epsilon_t := \langle \gamma_t g_t + \log(\widehat{q}_{t+1}) - \log(\widehat{q}_t), \widehat{q}_{t+1} - q \rangle$, which implies $\widehat{q}_{t+1} \in P_{\widehat{q}_t}^{\epsilon_t}(\gamma_t g_t)$. Let the bandwidth at step t satisfies*

$$h_t = O(1) m_t^{-1/(d+2\beta)},$$

one can guarantee that

$$\mathbb{E}_\theta[\epsilon_t|x_{[t-1]},\theta_{[t-1]}] \leqslant \quad O(1)\Delta C_k\left(\nu\mathcal{L}+\mu\right) m_t^{-\frac{\beta}{d+2\beta}}.$$

In addition, with probability at least $1-\delta$ *in* $\theta_t|x_{[t-1]},\theta_{[t-1]}$*, we have*

$$\epsilon_t \leqslant O(1)\Delta C_k\left(\nu\mathcal{L}+\mu\right)\left(m_t^{-\frac{\beta}{d+2\beta}}+\sqrt{\log\left(1/\delta\right)}m_t^{-\frac{1}{2}}\right)$$

where $O(1)$ *is some constant.*

Proof Note that since $\tilde{q}_t^+(\theta)=\tilde{q}_t(\theta)\exp(-\gamma_t g_t(\theta))/Z$, where $\tilde{q}_t(\theta)=\sum_{i=1}^{m_t}\alpha_i K_{h_t}(\theta-\theta_i)$, and $g_t(\theta)=\log(\tilde{q}_t)-\log(p)-N\log(p(x_t|\theta))$. By our assumption, we have $\tilde{q}_t\in C_{\mathcal{L}}^{\beta}(\Omega)$ and $\exp(-\gamma_t g_t)\in C_{\mathcal{L}}^{\beta}(\Omega)$; hence, $\tilde{q}_t^+\in C_{\mathcal{L}}^{\beta}(\Omega)$.

$$\begin{aligned}
\epsilon_t &:= \langle\gamma_t g_t+\log(\widehat{q}_{t+1})-\log(\widehat{q}_t),\widehat{q}_{t+1}-q\rangle\\
&= \langle\gamma_t g_t+\log(\widehat{q}_t^+)-\log(\widehat{q}_t),\widehat{q}_{t+1}-q\rangle+\langle\log\left(\widehat{q}_{t+1}\right)-\log(\tilde{q}_t^+),\widehat{q}_{t+1}-q\rangle\\
&\leqslant \Delta\left\|\widehat{q}_{t+1}-\tilde{q}_t^+\right\|_1\left\|\widehat{q}_{t+1}-q\right\|_\infty\\
&\leqslant \Delta\left\|\widehat{q}_{t+1}-\tilde{q}_t^+\right\|_1\left(\left\|\widehat{q}_{t+1}\right\|_\infty+\|q\|_\infty\right)\\
&\leqslant \Delta\left(\left\|\widehat{q}_{t+1}\right\|_\infty+C_\infty\right)\left\|\widehat{q}_{t+1}-\tilde{q}_t^+\right\|_1.
\end{aligned}$$

Based on the definition of $\widehat{q}_{t+1}$, we have

$$\begin{aligned}
\|\tilde{q}_t^+-\widehat{q}_{t+1}\|_1 &= \left\|\frac{1}{1-\zeta}\tilde{q}_{t+1}-\tilde{q}_t^+\right\|_1=\frac{1}{1-\zeta}\|\tilde{q}_{t+1}-\tilde{q}_t^++\zeta\tilde{q}_t^+\|_1\\
&\leqslant \frac{1}{1-\zeta}\|\tilde{q}_{t+1}-\tilde{q}_t^+\|_1+\frac{\zeta}{1-\zeta}\\
&= \|\tilde{q}_{t+1}-\tilde{q}_t^+\|_1+\zeta+o(\zeta+\|\tilde{q}_{t+1}-\tilde{q}_t^+\|_1).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\|\tilde{q}_t^+ - \widehat{q}_{t+1}\|_2 &= \left\| \frac{1}{1-\zeta}(\tilde{q}_{t+1} - \tilde{q}_t^+) + \frac{\zeta}{1-\zeta}\tilde{q}_t^+ \right\|_2 \\
&\leqslant \frac{1}{(1-\zeta)}\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_2 + \frac{\zeta}{(1-\zeta)}\|\tilde{q}_t^+\|_2 \\
&\leqslant (1+\zeta)\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_2 + \zeta\|\tilde{q}_t^+\|_2 + o(\zeta\|\tilde{q}_t^+\|_2 + \zeta\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_2).
\end{aligned}$$

Recall $\zeta = 1 - \int_\Omega \tilde{q}_{t+1} = \langle 1, \tilde{q}_t^+ - \tilde{q}_{t+1}\rangle \leqslant \|\tilde{q}_{t+1} - \tilde{q}_t^+\|_1$, we can simplify the L_1 and L_2 error as

$$\begin{aligned}
\|\widehat{q}_{t+1} - \tilde{q}_t^+\|_1 &= 2\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_1 + o(\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_1), \\
\|\tilde{q}_t^+ - \widehat{q}_{t+1}\|_2 &\leqslant \|\tilde{q}_{t+1} - \tilde{q}_t^+\|_2 + \|\tilde{q}_t^+\|_2\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_1 \\
&\quad + o(\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_2 + \|\tilde{q}_t^+\|_2\|\tilde{q}_{t+1} - \tilde{q}_t^+\|_1).
\end{aligned}$$

The last inequality for L_2 error comes from Jensen's inequality. We argue that $\|\widehat{q}_{t+1}\|_\infty$ is finite. Indeed,

$$\begin{aligned}
\|\widehat{q}_{t+1}\|_\infty &= \frac{1}{(1-\zeta)}\|\tilde{q}_{t+1}\|_\infty \leqslant (1+\zeta)\|\tilde{q}_{t+1}\|_\infty + o(\zeta\|\tilde{q}_{t+1}\|_\infty) \\
&= \|\tilde{q}_{t+1}\|_\infty + o(\|\tilde{q}_{t+1}\|_\infty) = \left\| \frac{\tilde{q}_t \exp(-\gamma_t g_t)}{Z} \right\|_\infty + o(\|\tilde{q}_{t+1}\|_\infty) \\
&\leqslant \exp(2\gamma_t M) \left\| \sum_i \alpha_i^{t+1} K_h(\theta - \theta_i) \right\|_\infty + o(\|\tilde{q}_{t+1}\|_\infty) \\
&\leqslant \exp(2\gamma_t M) C_k \|\alpha^{t+1}\|_\infty + o(\|\tilde{q}_{t+1}\|_\infty) \leqslant \exp(2\gamma_t M) C_k + o(\exp(2\gamma_t M) C_k).
\end{aligned}$$

Therefore, we have

$$\epsilon_t \leqslant 2\Delta(\exp(2\gamma_t M) C_K + C_\infty + o(\exp(2\gamma_t M) C_K)) \left\|\tilde{q}_{t+1} - \tilde{q}_t^+\right\|_1 + o\left(\left\|\tilde{q}_{t+1} - \tilde{q}_t^+\right\|_1\right).$$

Applying the result of Theorem 17 and Lemma 48 for $\widehat{q}_{t+1}$ and $\tilde{q}_t^+$ we have

$$\begin{aligned}
& \mathbb{E}_\theta[\epsilon_t|x_{[t-1]},\theta_{[t-1]}] \\
\leqslant\ & 2\Delta\left(\exp\left(2\gamma_t M\right)C_k + C_\infty + o\left(\exp\left(2\gamma_t M\right)C_k\right)\right)\bullet \\
& \left[\nu\mathcal{L}h^\beta + \frac{\mu}{\sqrt{m_t}h_t^{d/2}}\|\omega_t\sqrt{\tilde{q}_t}\|_1 + \frac{1}{\sqrt{m_t}}\|\omega_t\sqrt{\tilde{q}_t}\|_2 + o((m_t h_t^d)^{-\frac{1}{2}})\right] \\
& + o\left(\nu\mathcal{L}h_t^\beta + \frac{\mu}{\sqrt{m_t}h_t^{d/2}}\|\omega_t\sqrt{\tilde{q}_t}\|_1 + \frac{1}{\sqrt{m_t}}\|\omega_t\sqrt{\tilde{q}_t}\|_2\right)
\end{aligned}$$

Under the Assumptions, we already proved that $|\omega_t|_\infty \leqslant \exp(2\gamma_t M)$, hence, $\|\omega_t\sqrt{\tilde{q}_t}\|_2^2 \leqslant \exp(4\gamma_t M)$. Without loss of generality, we can assume $\int\sqrt{\tilde{q}_t(\theta)}d\theta \leqslant O(1)$ and $\gamma_t M \leqslant O(1)$ for all t, then we can simply write $\|\omega_t\sqrt{\tilde{q}_t}\|_1 \leqslant O(1)$ and $\|\omega_t\sqrt{\tilde{q}_t}\|_2^2 \leqslant O(1)$.

When $h_t = O(1)m_t^{-1/(d+2\beta)}$, the above result can be simplified as

$$\mathbb{E}_\theta[\epsilon_t|x_{[t-1]},\theta_{[t-1]}] \leqslant \quad O(1)\Delta C_k\left(\nu\mathcal{L}+\mu\right)m_t^{-\frac{\beta}{d+2\beta}}.$$

Similarly, combining the results of Corollary 50, we have with probability at least $1-\delta$,

$$\begin{aligned}
\epsilon_t \leqslant\ & 2\Delta\left(\exp\left(2\gamma_t M\right)C_k + C_\infty + o\left(\exp\left(2\gamma_t M\right)C_k\right)\right)\bullet \\
& \left[\nu\mathcal{L}h^\beta + \frac{\mu}{\sqrt{m_t}h_t^{d/2}}\|\omega_t\sqrt{\tilde{q}_t}\|_1 + \frac{1}{\sqrt{m_t}}\|\omega_t\sqrt{\tilde{q}_t}\|_2 + \frac{1}{\sqrt{m_t}}\sqrt{8B_1B_2\log(1/\delta)} + o((m_t h_t^d)^{-\frac{1}{2}})\right] \\
& + o\left(\nu\mathcal{L}h_t^\beta + \frac{\mu}{\sqrt{m_t}h_t^{d/2}}\|\omega_t\sqrt{\tilde{q}_t}\|_1 + \frac{1}{\sqrt{m_t}}\|\omega_t\sqrt{\tilde{q}_t}\|_2\right),
\end{aligned}$$

which leads to the lemma.

■

Our main Theorem 19 follows immediately by applying the results in the above lemma to Theorem 15.

Proof of Theorem 19 We first notice that

$$\begin{aligned}
\mathbb{E}[KL(q^*||\tilde{q}_T)] &= \mathbb{E}\left[\int q^*\log\frac{q^*}{\tilde{q}_T}d\theta\right] = \mathbb{E}\left[\int q^*\log\frac{q^*}{\widehat{q}_T}d\theta + \int q^*\log\frac{\widehat{q}_T}{\tilde{q}_T}d\theta\right] \\
&= \mathbb{E}[KL(q^*||\widehat{q}_T)] + \mathbb{E}\left[\int q^*\log\frac{\widehat{q}_T}{\tilde{q}_T}d\theta\right].
\end{aligned}$$

For the second term,

$$\begin{aligned}\mathbb{E}\left[\int q^* \log \frac{\widehat{q}_T}{\tilde{q}_T} d\theta\right] &= \mathbb{E}\left[\langle q^*, \log \frac{\frac{1}{1-\zeta_T}\tilde{q}_T}{\tilde{q}_T}\rangle\right] = \mathbb{E}\left[\langle q^*, -\log(1-\zeta_T)\right] \\ &= \mathbb{E}[-\log(1-\zeta_T)] \leqslant \zeta_T + o(\zeta_T) \\ &\leqslant \mathbb{E}\|\tilde{q}_T - \tilde{q}_{T-1}^+\|_1 + o(\mathbb{E}\|\tilde{q}_T - \tilde{q}_{T-1}^+\|_1)\end{aligned}$$

By Theorem 17 and setting $h_t = O(1)m_t^{-1/(d+2\beta)}$, we achieve the error bound

$$\mathbb{E}\left[\int q^* \log \frac{\widehat{q}_T}{\tilde{q}_T} d\theta\right] \leqslant \mathcal{C}_2 m_t^{-\frac{\beta}{d+2\beta}},$$

where $\mathcal{C}_2 := O(1)(\mu + \nu\mathcal{L})$.

Meanwhile, from Lemma 53, we have

$$\mathbb{E}_\theta[\epsilon_t | x_{[t-1]}, \theta_{[t-1]}] \leqslant \mathcal{C}_1 m_t^{-\beta/(d+2\beta)},$$

where $\mathcal{C}_1 := O(1)\Delta C_k\left(\nu\mathcal{L} + \mu\right)$. Expanding the result from Theorem 15, it follows that

$$\mathbb{E}_{x,\theta}[KL(q^*||\widehat{q}_{t+1})] \leqslant (1-\gamma_t)\mathbb{E}_{x,\theta}[KL(q^*||\widehat{q}_t)] + \mathcal{C}_1 m_t^{-\beta/(d+2\beta)} + \frac{\gamma_t^2}{2}M^2,$$

by setting $\gamma_t = \frac{2}{t+1}$. The above recursion leads to the convergence result for the second term,

$$\mathbb{E}[KL(q^*||\widehat{q}_T)] \leqslant \frac{2\max\left\{D_1, M^2\right\}}{T} + \mathcal{C}_1 \frac{\sum_{t=1}^T t^2 m_t^{-\frac{\beta}{d+2\beta}}}{T^2}.$$

Combine these two results, we achieve the desired result

$$\mathbb{E}[KL(q^*||\tilde{q}_T)] \leqslant \frac{2\max\left\{D_1, M^2\right\}}{T} + \mathcal{C}_1 \frac{\sum_{t=1}^T t^2 m_t^{-\frac{\beta}{d+2\beta}}}{T^2} + \mathcal{C}_2 m_t^{-\frac{\beta}{d+2\beta}}.$$

■

Remark: The convergence in terms of KL-divergence is measuring the entire density and much more stringent compared to integral approximation. For the last iterate, an overall

$O(\frac{1}{T})$ convergence rate can be achieved when $m_t = O(t^{2+d/\beta})$. Similar to Lemma 42, with Pinsker's inequality, we could easily obtain the the rate of convergence in terms of integral approximation from Theorem 19. After T steps, in general cases, the PMD algorithm converges in terms of integral approximation in rate $O(1/\sqrt{T})$ by choosing $O(1/t)$-decaying stepsizes and $O(t^{2+\frac{d}{\beta}})$-growing samples.

B.6 Derivation Details for Sparse Gaussian Processes and Latent Dirichlet Allocation

We apply the Particle Mirror Descent algorithm to sparse Gaussian processes and latent Dirichlet allocation. For these two models, we decompose the latent variables and incorporate the structure of posterior into the algorithm. The derivation details are presented below.

B.6.1 Sparse Gaussian Processes

Given data $X = \{x_i\}_{i=1}^n, \quad x_i \in \mathbb{R}^{d\times 1}$ and $y = \{y_i\}_{i=1}^n$. The sparse GP introduce a set of inducing variables, $Z = \{z_i\}_{i=1}^m, \quad z_i \in \mathbb{R}^{d\times 1}$ and the model is specified as

$$\begin{aligned} p(y_n|\mathbf{u}, Z) &= \mathcal{N}(y_n|K_{nm}K_{mm}^{-1}\mathbf{u}, \tilde{K}) \\ p(\mathbf{u}|Z) &= \mathcal{N}(\mathbf{u}|\mathbf{0}, K_{mm}). \end{aligned}$$

where $K_{mm} = [k(z_i, z_j)]_{i,j=1,\ldots,m}$, $K_{nm} = [k(x_i, z_j)]_{i=1,\ldots,n;j=1,\ldots,m}$. For different $\tilde{K}$, there are different sparse approximations for GPs. Please refer [128] for details. We test algorithms on the sparse GP model with $\tilde{K} = \beta^{-1}I$. We modify the stochastic variational inference for Gaussian processes [123] for this model. We also apply our algorithm on the same model. However, it should be noticed that our algorithm could be easily extended to other sparse approximations [128].

We treat the inducing variables as the latent variables with uniform prior in sparse Gaussian processes. Then, the posterior of $Z, \mathbf{u}$ could be thought as the solution to the

optimization problem

$$\min_{q(Z,\mathbf{u})} \int q(Z,\mathbf{u}) \log \frac{q(Z,\mathbf{u})}{p(Z)p(\mathbf{u})} \mathbf{u} dZ - \sum_{i=1}^{n} \int q(Z,\mathbf{u}) \log p(y_i|x_i,\mathbf{u},Z) d\mathbf{u} dZ \tag{B.9}$$

The stochastic gradient of Eq.(B.9) w.r.t. $q(Z,\mathbf{u})$ will be

$$g(q(Z,\mathbf{u})) = \frac{1}{n} \log q(Z,\mathbf{u}) - \frac{1}{n} \log p(Z)p(\mathbf{u}) - \log p(y_i|x_i,\mathbf{u},Z)$$

and therefore, the prox-mapping in t-step is

$$\min_{q(Z,\mathbf{u})} \int q(Z,\mathbf{u}) \log \frac{q(Z,\mathbf{u})}{q_t(Z,\mathbf{u})^{1-\gamma_t/n} p(Z,\mathbf{u})^{\gamma_t/n}} \mathbf{u} dZ - \gamma_t \int q(Z,\mathbf{u}) \log p(y_i|x_i,\mathbf{u},Z) d\mathbf{u} dZ$$

which could be re-written as

$$\begin{aligned} & \min_{q(Z)q(\mathbf{u}|Z)} \int q(Z) \Bigg\{ \log \frac{q(Z)}{q_t(Z)^{1-\gamma_t/n} p(Z)^{\gamma_t/n}} \\ + & \underbrace{\int q(\mathbf{u}|Z) \left[\log \frac{q(\mathbf{u}|Z)}{q_t(\mathbf{u}|Z)^{1-\gamma_t/n} p(\mathbf{u}|Z)^{\gamma_t/n}} - \gamma_t \log p(y_i|x_i,\mathbf{u},Z) \right] d\mathbf{u}}_{L(q(\mathbf{u}|Z))} \Bigg\} dZ \end{aligned}$$

We update $q_{t+1}(\mathbf{u}|Z)$ to be the optimal of $L(q(\mathbf{u}|Z))$ as

$$\begin{aligned} q_{t+1}(\mathbf{u}|Z) &\propto q_t(\mathbf{u}|Z)^{1-\gamma_t/n} p(\mathbf{u}|Z)^{\gamma_t/n} p(y_i|x_i,\mathbf{u},Z)^{\gamma_t} \\ &= \mathcal{N}(\mathbf{u}|m_t, \delta_t^{-1})^{1-\gamma_t/n} \mathcal{N}(\mathbf{u}|\mathbf{0}, K_{mm})^{\gamma_t/n} \mathcal{N}(y_i|K_{im}K_{mm}^{-1}\mathbf{u}, \Gamma)^{\gamma_t} \\ &= \mathcal{N}(\mathbf{u}|m_{t+1}, \delta_{t+1}^{-1}) \end{aligned}$$

where $\Gamma = diag(\tilde{K}_{ii} - Q_{ii}) + \beta^{-1} I,\ Q_{ii} = K_{im}K_{mm}^{-1}K_{mi}$,

$$\begin{aligned} \delta_{t+1} &= (1-\gamma_t/n)\delta_t + \gamma_t/n K_{mm}^{-1} + \gamma_t K_{im} K_{mm}^{-1} \Gamma^{-1} K_{mm}^{-1} K_{mi} \\ m_{t+1} &= \delta_{t+1}^{-1} \Big((1-\gamma_t/n)\delta_t^{-1} m_t + \gamma_t/n K_{mm}^{-1} m_0 + \gamma_t K_{mm}^{-1} K_{mi} \Gamma^{-1} y \Big) \end{aligned}$$

Plug this into the $L(q(\mathbf{u}|Z))$, we have

$$\begin{aligned} L(q(u|Z)) &= \int q(\mathbf{u}|Z)\left[\log \frac{q(\mathbf{u}|Z)}{q_t(\mathbf{u}|Z)^{1-\gamma_t/n}p(\mathbf{u}|Z)^{\gamma_t/n}} - \gamma_t \log p(y_i|x_i,\mathbf{u},Z)\right]d_{\mathbf{u}} \\ &= -\log \tilde{p}(y_i|x_i,Z) \end{aligned}$$

where

$$\begin{aligned} \tilde{p}(y_i|x_i,Z) &= \int q_t(\mathbf{u}|Z)^{1-\gamma_t/n}p(\mathbf{u}|Z)^{\gamma_t/n}p(y_i|x_i,\mathbf{u},Z)^{\gamma_t}d\mathbf{u} \\ &= \int \mathcal{N}(\mathbf{u}|m_t,\delta_t^{-1})^{1-\gamma_t/n}\mathcal{N}(\mathbf{u}|0,K_{mm})^{\gamma_t/n}\mathcal{N}(y_i|K_{im}K_{mm}^{-1}\mathbf{u},\Gamma)^{\gamma_t}d\mathbf{u} \\ &= \mathcal{N}(y_i|K_{im}K_{mm}^{-1}c,\Sigma) \end{aligned}$$

where

$$\begin{aligned} \bar{\delta}_{t+1} &= (1-\gamma_t/n)\delta_t + \gamma_t/nK_{mm}^{-1} \\ c &= \bar{\delta}_{t+1}^{-1}\left((1-\gamma_t/n)\delta_t m_t + \gamma_t/nK_{mm}^{-1}m_0\right) \\ \Sigma &= K_{im}K_{mm}^{-1}\bar{\delta}_{t+1}^{-1}K_{mm}^{-1}K_{mi} + \frac{1}{\gamma_t}\Gamma \end{aligned}$$

Solve

$$\min_{q(Z)} \int q(Z)\log\frac{q(Z)}{q_t(Z)^{1-\gamma_t/n}p(Z)^{\gamma_t/n}}dZ - \int q(Z)\log\tilde{p}(y_i|x_i,Z)dZ$$

will result the update rule for $q(Z)$,

$$q_{t+1}(Z) \propto q_t(Z)^{1-\gamma_t/n}p(Z)^{\gamma_t/n}\tilde{p}(y_i|x_i,Z)$$

We approximate the $q(Z)$ with particles, i.e., $q(Z) = \sum_{j=1}^{l} w^j\delta(Z^j)$. The update rule for w^j is

$$w_{t+1}^j = \frac{w_t^j \exp(-\gamma_t/n\log(w_t^j) + \gamma_t/n\log p(Z^j) + \log\tilde{p}(y_i|x_i,Z^j))}{\sum_j^l w_t^j \exp(-\gamma_t/n\log(w_t^j) + \gamma_t/n\log p(Z^j) + \log\tilde{p}(y_i|x_i,Z^j))}$$

B.6.2 Latent Dirichlet Allocations

In LDA, the topics $\Phi \in \mathbb{R}^{K\times W}$ are K distributions on the words W in the text corpora. The text corpora contains D documents, the length of the d-th document is N_d. The document is modeled by a mixture of topics, with the mixing proportion $\theta_d \in \mathbb{R}^{1\times K}$. The words generating process for X_d is following: first drawing a topic assignment z_{dn}, which is 1-by-K indicator vector, *i.i.d.*from θ_d for word x_{dn} which is 1-by-W indicator vector, and then drawing the word x_{dn} from the corresponding topic $\Phi_{z_{dn}}$. We denote $z_d = \{z_{dn}\}_{n=1}^{N_d} \in \mathbb{R}^{N_d\times K}$, $x_d = \{x_{dn}\}_{n=1}^{N_d} \in \mathbb{R}^{N_d\times W}$ and $X = \{x_d\}_{d=1}^{D}$,$Z = \{Z_d\}_{d=1}^{D}$. Specifically, the joint probability is

$$\begin{aligned} p(x_d, z_d, \theta_d, \Phi) &= p(x_d|z_d, \Phi)p(z_d|\theta_d)p(\theta_d)p(\Phi) \\ p(x_d|z_d, \Phi) &= \prod_{n=1}^{N_d}\prod_{w=1}^{W}\prod_{k=1}^{K}\Phi_{kw}^{z_{dnk}x_{dnw}} \\ p(z_d|\theta_d) &= \prod_{n=1}^{N_d}\prod_{k=1}^{K}\theta_{dk}^{z_{dnk}} \end{aligned} \tag{B.10}$$

The $p(\Phi)$ and $p(\theta)$ are the priors for parameters, $p(\theta_d|\alpha) = \frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K}\prod_k^K \theta_{dk}^{\alpha-1}$ and $p(\Phi|\beta_0) = \prod_k^K \frac{\Gamma(W\beta_0)}{\Gamma(\beta_0)^W}\prod_w^W \Phi_{wk}^{\beta_0-1}$, both are Dirichlet distributions.

We incorporate the special structure into the proposed algorithm. Instead of modeling the $p(\Phi)$ solely, we model the $Z = \{Z\}_{d=1}^{D}$ and Φ together as $q(Z, \Phi)$. Based on the model, given Z, the $q(\Phi|Z)$ will be Dirichlet distribution and could be obtained in closed-form.

The posterior of Z, Φ is the solution to

$$\min_{q(Z,\Phi)} \frac{1}{D}\int q(Z,\Phi)\log\frac{q(Z,\Phi)}{p(Z|\alpha)p(\Phi|\beta)}dZd\Phi - \frac{1}{D}\sum_{d=1}^{D}\int q(Z,\Phi)\log p(x_d|z_d,\Phi)dZd\Phi$$

We approximate the finite summation by expectation, then the objective function becomes

$$\min_{q(Z,\Phi)} \frac{1}{D}\int q(Z,\Phi)\log\frac{q(Z,\Phi)}{p(Z|\alpha)p(\Phi|\beta)}dZd\Phi - \mathbb{E}_x\left[\int q(Z,\Phi)\log p(x_d|z_d,\Phi)dZd\Phi\right] \tag{B.11}$$

We approximate the $q(Z) \approx \sum_{i=1}^{m} w^i\delta(Z^i)$ by particles, and therefore, $q(Z,\Phi) \approx \sum_{i=1}^{m} w^i P(\Phi|Z^i)$

where $P(\Phi|Z^i)$ is the Dirichlet distribution as we discussed. It should be noticed that from the objective function, we do not need to instantiate the z_d until we visit the x_d. By this property, we could first construct the particles $\{Z^i\}_{i=1}^m$ 'conceptually' and assign the value to $\{z_d^i\}_{i=1}^m$ when we need it. The gradient of Eq.(B.11) w.r.t. $q(\Phi, Z)$ is

$$g(q(Z,\Phi)) = \frac{1}{D}\log q(Z,\Phi) - \frac{1}{D}\log p(\Phi)p(Z) - \mathbb{E}_x[\log p(x_d|\Phi, z_d)]$$

Then, the SGD prox-mapping is

$$\begin{aligned}\min_{q(Z,\Phi)} \quad & \int q(Z,\Phi)\log\frac{q(Z,\Phi)}{q_t(Z,\Phi)} \\ & +\gamma_t\int q(Z,\Phi)\Big[\log q_t(Z,\Phi)/D - \log p(\Phi)p(Z)/D - \log p(x_d|\Phi,z_d)\Big]dZd\Phi\end{aligned}$$

We rearrange the prox-mapping,

$$\begin{aligned}\min_{q(Z)q(\Phi|Z)} \quad & \int q(Z)q(\Phi|Z)\log\frac{q(Z)q(\Phi|Z)}{q_t(Z)^{1-\gamma_t/D}q_t(\Phi|Z)^{1-\gamma_t/D}} \\ - \quad & \gamma_t\int q(Z)q(\Phi|Z)\Big[\log p(\Phi)p(Z)/D + \log p(x_d|\Phi,z_d)\Big]dZd\Phi\end{aligned}$$

$$\begin{aligned}\min_{q(Z)q(\Phi|Z)} \quad & \int q(Z)\Big\{\log\frac{q(Z)}{q_t(Z)^{1-\gamma_t/D}p(Z)^{\gamma_t/D}} \\ + \quad & \underbrace{\int q(\Phi|Z)\Big[\log\frac{q(\Phi|Z)}{q_t(\Phi|Z)^{1-\gamma_t/D}p(\Phi)^{\gamma_t/D}} - \gamma_t\log p(x_d|\Phi,z_d)\Big]d\Phi}_{L(q(\Phi|Z))}\Big\}dZ\end{aligned}$$

The stochastic functional gradient update for $q(\Phi|Z^i)$ is

$$q_{t+1}(\Phi|Z^i) \propto q_t(\Phi|Z^i)^{1-\gamma_t/D}p(\Phi)^{\gamma_t/D}p(x_d|\Phi,z_d)^{\gamma_t}$$

Let $q_t(\Phi|Z^i) = \mathcal{D}ir(\beta_t^i)$, then, the $q_{t+1}(\Phi|Z^i)$ is also Dirichlet distribution

$$q_{t+1}(\Phi|Z^i) \propto \mathcal{D}ir(\beta_t^i)^{1-\tilde{\gamma}_t}\mathcal{D}ir(\beta_0)^{\tilde{\gamma}_t}\Big(\prod_k\prod_w \Phi_{kw}^{\sum_n^{N_d}\delta(z_{dnk}=1,x_{dnw}=1)}\Big)^{D\tilde{\gamma}_t} = \mathcal{D}ir(\beta_{t+1}^i)$$

where $\tilde{\gamma}_t = \gamma_t / D$ and

$$[\beta^i_{t+1}]_{kw} = (1-\tilde{\gamma}_t)[\beta^i_t]_{kw} + \tilde{\gamma}_t\beta_0 + D\tilde{\gamma}_t \sum_n^{N_d} \delta(z_{dnk}=1, x_{dnw}=1).$$

In mini-batch setting, the updating will be

$$[\beta^i_{t+1}]_{kw} = (1-\tilde{\gamma}_t)[\beta^i_t]_{kw} + \tilde{\gamma}_t\beta_0 + \frac{D}{B}\tilde{\gamma}_t \sum_{d=1}^{B}\sum_n^{N_d} \delta(z_{dnk}=1, x_{dnw}=1).$$

Plug the $q_{t+1}(\Phi|Z^i)$ into prox-mapping, we have

$$\begin{aligned} L(q(\Phi|Z)) &= \int q(\Phi|Z)\bigg[\log\frac{q(\Phi|Z)}{q_t(\Phi|Z)^{1-\tilde{\gamma}_t}p(\Phi)^{\tilde{\gamma}_t}} - D\tilde{\gamma}_t \log p(x_d|\Phi, z_d)\bigg]d\Phi \\ &= -\log\tilde{p}(x_d|z_d, Z) \end{aligned}$$

where $\tilde{p}(x_d|z_d, Z^i) = \int_\Phi q_t(\Phi|Z^i)^{1-\tilde{\gamma}_t}p(\Phi)^{\tilde{\gamma}_t}p(x_d|\Phi, z_d)^{D\tilde{\gamma}_t}d\Phi$ which have closed-form

$$\begin{aligned} \tilde{p}(x_d|z_d, Z^i) &= \int_\Phi q_t(\Phi|Z^i)^{1-\tilde{\gamma}_t}p(\Phi)^{\tilde{\gamma}_t}p(x_d|\Phi, z_d)^{D\tilde{\gamma}_t}d\Phi \\ &= \int \mathcal{D}ir(\beta^i_t)^{1-\tilde{\gamma}_t}\mathcal{D}ir(\beta^i_0)^{\tilde{\gamma}_t}\bigg(\prod_k\prod_w \Phi_{kw}^{\sum_n^{N_d}\delta(z_{dnk}=1,x_{dnw}=1)}\bigg)^{D\tilde{\gamma}_t}d\Phi \\ &= \prod_k\bigg(\frac{\Gamma(\sum_w^W[\beta^i_t]_{kw})}{\prod_w\Gamma([\beta^i_t]_{kw})}\bigg)^{1-\tilde{\gamma}_t}\bigg(\frac{\Gamma(W\beta_0)}{\Gamma(\beta_0)^W}\bigg)^{\tilde{\gamma}_t}\frac{\prod_w\Gamma([\beta^i_{t+1}]_{kw})}{\Gamma(\sum_w[\beta^i_{t+1}]_{kw})} \end{aligned}$$

and

$$\begin{aligned} \log\tilde{p}(x_d|z_d, Z^i) \propto \sum_k\bigg((1-\tilde{\gamma}_t)\log\Gamma(\sum_w^W[\beta^i_t]_{kw}) + \sum_w\log\Gamma([\beta^i_{t+1}]_{kw}) \\ - \log\Gamma(\sum_w[\beta^i_{t+1}]_{kw}) - (1-\tilde{\gamma}_t)\sum_w\log\Gamma([\beta^i_t]_{kw})\bigg) \end{aligned}$$

Then, we could update $q_t(Z) = \sum_i^m w^i\delta(Z^i)$ by

$$q_{t+1}(Z^i) \propto q_t(Z^i)\exp\bigg(-\frac{\gamma_t}{D}\log q_t(Z^i) + \frac{\gamma_t}{D}\log p(Z^i|\alpha) + \log\tilde{p}(x_d|z_d, Z^i)\bigg)$$

If we set $\alpha = 1$, $p(Z^i)$ will be uniformly distributed which has no effect to the update. For general setting, to compute $\log p(Z^i|\alpha)$, we need prefix all the $\{z^i_d\}_{d=1}^D$. However, when D

is huge, the second term will be small and we could ignore it approximately.

Till now, we almost complete the algorithm except the how to assign z_d when we visit x_d. We could assign the z_d randomly. However, considering the requirement for the z_d^i assignment that the $q(z_d^i|Z_{\backslash d}^i) > 0$, which means the assignment should be consistent, an better way is using the average or sampling proportional to $\int p(x_d|\Phi, z_d)q_t(\Phi|Z^i)p(z_d|Z_{1...,d-1}^i)d\Phi$ where $p(z_d|Z_{1...,d-1}^i) = \int p(z_d|\alpha)p(\alpha|Z_{1...,d-1}^i)d\alpha$, or $\int p(x_d|\Phi, z_d)q_t(\Phi|Z^i)p(z_d|\alpha)d\Phi$.

B.7 More Related Work

Besides the most related two inference algorithms we discussed in Section (4.5), *i.e.*, stochastic variational inference [106] and static sequential Monte Carlo [91, 92], there are several other inference algorithms connect to the PMD from algorithm, stochastic approximation, or representation aspects, respectively.

From algorithmic aspect, our algorithm scheme shares some similarities to annealed importance sampling (AIS) [116] in the sense that both algorithms are sampling from a series of densities and reweighting the samples to approximate the target distribution. The most important difference is the way to construct the intermediate densities. In AIS, the density at each iteration is a weighted product of the joint distribution of *all the data* and a *fixed* proposal distribution, while the densities in PMD are a weighted product of *previous step solution* and the stochastic functional gradient on *partial data*. Moreover, the choice of the temperature parameter (fractional power) in AIS is heuristic, while in our algorithm, we have a principle way to select the stepsize with quantitative analysis. The difference in intermediate densities results the sampling step in these two algorithms is also different: the AIS might need MCMC to generate samples from the intermediate densities, while we only samples from a KDE which is more efficient. These differences make our method could handle large-scale dataset while AIS cannot.

Sequential Monte-Carlo sampler [115] provides a unified view of SMC in Bayesian inference by adopting different forward/backward kernels, including the variants proposed in [91, 92] as special cases. There are subtle and important differences between the PMD and the SMC samplers. In the SMC samplers, the introduced finite forward/backward

Markov kernels are used to construct a distribution over the auxiliary variables. To make the SMC samplers valid, it is required that the marginal distribution of the constructed density by integrating out the auxiliary variables must be the *exact* posterior. However, there is no such requirement in PMD. In fact, the PMD algorithm only *approaches* the posterior with controllable error by iterating the dataset *many times*. Therefore, although the proposed PMD and the SMC sampler bare some similarities operationally, they are essentially different algorithms.

Stochastic approximation becomes a popular trick in extending the classic Bayesian inference methods to large-scale datasets recently. Besides stochastic variational inference, which incorporates stochastic gradient descent into variational inference, the stochastic gradient Langevin dynamics (SGLD) [93] and its derivatives [95, 96, 97] combine ideas from stochastic optimization and Hamiltonian Monte Carlo sampling. Although both PMD and the SGLD use the stochastic gradient information to guide next step sampling, the optimization variable in these two algorithms are different which results the completely different updates and properties. In PMD, we directly update the density utilizing *functional gradient in density space*, while the SGLD perturbs the *stochastic gradient in parameter space*. Because of the difference in optimization variables, the mechanism of these algorithms are totally different. The SGLD generates a trajectory of *dependent* samples whose stationary distribution approximates the posterior, the PMD keeps an approximation of the posterior represented by *independent* particles or their weighted kernel density estimator. In fact, their different properties we discussed in Table 4.1 solely due to this essential difference. After the PMD has been developed, the Stein variational gradient descent (SVGD) [121, 220] extends the functional gradient technique to the optimization problem (4.2) for the posterior family which is constructed by flows transformed through RKHS functions, resulting an interactive sampling algorithm. [220] justifies the SVGD as gradient flows and characterizes its asymptotic property, however, the convergence rate of SVGD is not clear yet. Algorithmically, comparing to PMD which can increase the sample size arbitrarily, the SVGD needs to fix the sample size at first step and cannot be changed during the whole algorithm.

A number of generalized variational inference approaches are proposed trying to relax

the constraints on the density space with flexible densities. Nonparametric density family is a natural choice[1]. [119] and [118, 120] extend the belief propagation algorithm with nonparametric models by kernel embedding and particle approximation, respectively. The most important difference between these algorithms and PMD is that they originate from different sources and are designed for different settings. Both the kernel BP [119] and particle BP [118, 120] are based on belief propagation optimizing *local objective* and designed for the problem with *one sample* X in which observations are highly dependent, while the PMD is optimizing the *global objective*, therefore, more similar to mean-field inference, for the inference problems with *many i.i.d.* samples.

After the comprehensive review about the similarities and differences between PMD and the existing related approximate Bayesian inference methods from algorithm, stochastic approximation and representation perspectives, we can see the position of the proposed PMD clearly. The PMD connects variation inference and Monte Carlo approximation, which seem two orthogonal paradigms in approximate Bayesian inference, and achieves a balance in trade-off between efficiency, flexibility and provability.

B.8 Additional Experiments

B.8.1 Sparse Gaussian Processes

1D Synthetic Dataset

We test the proposed algorithm on 1D synthetic data. The data are generated by

$$y = 3x^2 + (\sin(3.53\pi x) + \cos(7.7\pi x)) \exp(-1.6\pi|x|) + 0.1e$$

where $x \in [-0.5, 0.5]$ and $e \sim \mathcal{N}(0, 1)$. The dataset contains 2048 observations which is small enough to run the exact GP regression. We use Gaussian RBF kernel in Gaussian processes and sparse Gaussian processes. Since we are comparing different inference algorithms on the same model, we use the same hyperparameters for all the inference algorithms. We set the kernel bandwidth σ to be 0.1 times the median of pairwise distances between data points

[1] Although [117, 108] named their methods as "nonparametric" belief propagation and "nonparametric" variational inference, they indeed use mixture of Gaussians, which is still a parametric model.

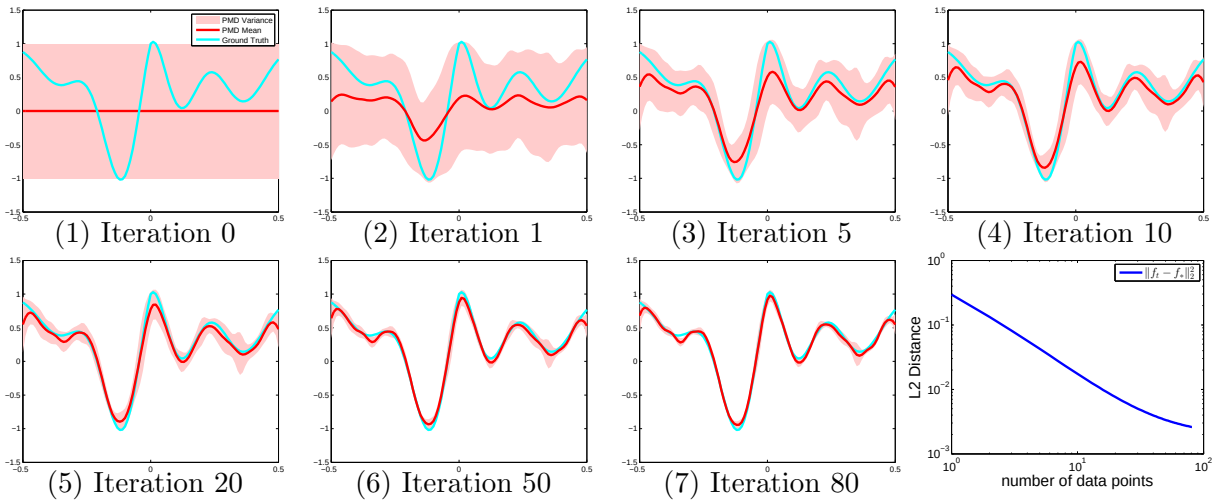


Figure B.1: Visualization of posterior prediction distribution. The red curve is the mean function and the pale red region is the variance of the posterior. The cyan curve is the ground truth. The last figure shows convergence of the posterior mean to the ground truth.

(median trick), and $\beta^{-1} = 0.001$. We set the stepsize in the form of $\frac{\eta}{n_0+\sqrt{t}}$ for both PMD and SVI and the batch size to be 128. Figure. B.1 illustrates the evolving of the posterior provided by PMD with 16 particles and 128 inducing variables when the algorithms visit more and more data. To illustrate the convergence of the posterior provided by PMD, we initialize the $\mathbf{u} = 0$ in PMD. Later, we will see we could make the samples in PMD more efficient.

B.8.2 Latent Dirichlet Allocation

The log-perplexity was estimated using the methods discussed in [124] on a separate holdout set with 1000 documents. For a document x_d in holdout set, the perplexity is computed by

$$\text{perp}(x_d|X,\alpha,\beta) = \exp\left(-\frac{\sum_{n=1}^{N_d}\log p(x_{dn}|X,\alpha,\beta)}{N_d}\right)$$

where

$$p(x_{dn}|X,\alpha,\beta) = \mathbb{E}_{\theta_d,\Phi}\left[\theta_d^\top \Phi_{\cdot,x_{dn}}\right]. \tag{B.12}$$

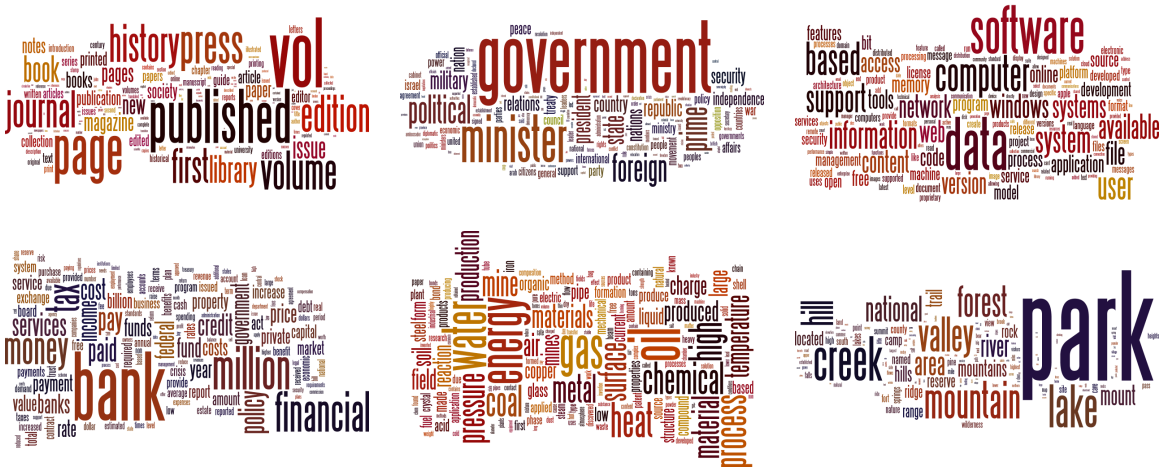


Figure B.2: Several topics learnd by LDA with PMD.

We separate the documents in testing set into two non-overlapped parts, $x_d^{\text{estimation}}$ and $x_d^{\text{evaluation}}$. We first evaluate the θ_d based on the $x_d^{\text{estimation}}$. For different inference methods, we use the corresponding strategies in learning algorithm to obtain the distribution of θ_d based on $x_d^{\text{estimation}}$. We evaluate $p(x_{dn}|X, \alpha, \beta)$ on $x_d^{\text{evaluation}}$ with the obtained distribution of θ_d. Specifically,

$$p(x_{dn}^{\text{evaluation}}|X, \alpha, \beta) = \mathbb{E}_{\Phi|X,\beta}\mathbb{E}_{\theta_d^{\text{evaluation}}|\Phi,\alpha,x_d^{\text{estimation}}}\left[\theta_d^\top \Phi_{\cdot,x_{dn}}\right]$$

For PMD, SMC and stochastic Langevin dynamics,

$$\theta_{dk}^{\text{evaluation}} = \frac{\sum_{n=1}^{N_d^{\text{estimation}}} \delta(z_{dnk}^{\text{estimation}} = 1) + \alpha}{N_d^{\text{estimation}} + K\alpha}$$

For stochastic variational inference, $q(\theta_d)$ is updated as in the learning procedure.

We illustrate several topics learned by LDA with our algorithm in Figure.B.2.

APPENDIX C
PROOF DETAILS AND EXTENSION OF DUAL EMBEDDING IN CHAPTER 5

C.1 Interchangeability Principle and Dual Continuity

Lemma 20 *Let ξ be a random variable on Ξ and assume for any $\xi \in \Xi$, function $g(\cdot, \xi) : \mathbb{R} \to (-\infty, +\infty)$ is a proper and upper semicontinuous concave function. Then*

$$\mathbb{E}_\xi[\max_{u\in\mathbb{R}} g(u,\xi)] = \max_{u(\cdot)\in\mathcal{G}(\Xi)} \mathbb{E}_\xi[g(u(\xi),\xi)].$$

where $\mathcal{G}(\Xi) = \{u(\cdot) : \Xi \to \mathbb{R}\}$ is the entire space of functions defined on support Ξ.

Proof First of all, by assumption of concavity and upper-semicontinuity, we know that for any $\xi \in \Xi$, there exists a maximizer for $\max_u g(u,\xi)$; let us denote as u^*_ξ. We can therefore define a function $u^*(\cdot) : \mathcal{X} \to \mathbb{R}$ such that $u^*(\xi) = u^*_\xi$, and thus, $u^*(\cdot) \in \mathcal{G}(\Xi)$. Hence,

$$\mathbb{E}_\xi[\max_{u\in\mathbb{R}} g(u,\xi)] = \mathbb{E}_\xi[g(u^*(\xi),\xi)] \leqslant \max_{u(\cdot)\in\mathcal{G}(\mathcal{X})} \mathbb{E}_\xi[g(u(\xi),\xi)].$$

On the other hand, clearly, for any $u(\cdot) \in \mathcal{G}(\Xi)$ and $\xi \in \Xi$, $g_\xi(u(\xi),\xi) \leqslant \max_{u\in\mathbb{R}} g(u,\xi)$. Hence, $\mathbb{E}_\xi[g(u(\xi),\xi)] \leqslant \mathbb{E}_\xi[\max_{u\in\mathbb{R}} g(u,\xi)]$, for any $u(\cdot) \in \mathcal{G}(\Xi)$. This further implies that

$$\max_{u(\cdot)\in\mathcal{G}(\Xi)} \mathbb{E}_\xi[g(u(\xi),\xi)] \leqslant \mathbb{E}_\xi[\max_{u\in\mathbb{R}} g(u,\xi)].$$

Combining these two facts leads to the statement in the lemma. ■

Proposition 21 *Suppose both $f(z,x)$ and $p(z|x)$ are continuous in x for any z,*

(1) (Discrete case) If the loss function $\ell_y(v)$ is continuously differentiable in v for any $y \in \mathcal{Y}$, then $u^*(x,y)$ is unique and continuous in x for any $y \in \mathcal{Y}$;

(2) (Continuous case) If the loss function $\ell_y(v)$ is continuously differentiable in (v, y), then $u^*(x, y)$ is unique and continuous in (x, y) on $\mathcal{X} \times \mathcal{Y}$.

Proof The continuity properties of optimal dual function follows directly from the fact that $u^*(x, y) \in \partial \ell_y(\mathbb{E}_{z|x}[f(z, x)])$. In both cases, for any $y \in \mathcal{Y}$, $\ell_y(\cdot)$ is differentiable. Hence $u^*(x, y) = \ell'_y(\int f(z, x)p(z|x)dz)$ is unique. Since $f(z, x)$ and $p(z|x)$ is continuous in x for any z, then $\mathbb{E}_{z|x}[f(z, x)]$ is continuous in x. Since for any $y \in \mathcal{Y}$, $\ell'_y(\cdot)$ is continuous, the composition $u^*(x, y)$ is therefore continuous in x as well. Moreover, if $\ell'_y(\cdot)$ is also continuous in $y \in \mathcal{Y}$, then the composition $u^*(x, y)$ is continuous in (x, y). ■

Indeed, suppose $\ell_y(\cdot)$ is uniformly L-Lipschitz differentiable for any $y \in \mathcal{Y}$, $f(z, x)$ is uniformly M_f-Lipschitz continuous in x for any z, $p(z|x)$ is M_p-Lipschitz continuous in x. Then

$$\begin{aligned}
|u^*(x_1, y) - u^*(x_2, y)| &= \left| \ell'_y \left(\int f(z, x_1)p(z|x_1)dz \right) - \ell'_y \left(\int f(z, x_2)p(z|x_2)dz \right) \right| \\
&\leqslant L \int |f(z, x_1)p(z|x_1) - f(z, x_2)p(z|x_2)|dz \\
&\leqslant L \int |f(z, x_1) - f(z, x_2)|p(z|x_1)dz \\
&+ L \int |f(z, x_2)| \cdot |p(z|x_1) - p(z|x_2)|dz \\
&\leqslant LM_f|x_1 - x_2| + LM_p|x_1 - x_2| \sup_x \int |f(z, x)|dz
\end{aligned}$$

If for any $f(z, x)$ is Lebesgue integrable and $\int |f(z, x)|dz$ is uniformly bounded, then $u^*(x, y)$ is also Lipschitz-continuous for any $y \in \mathcal{Y}$. Moreover, if in addition, $\ell_y(v)$ is also Lipschitz differentiable in (v, y), then $u^*(x, y)$ is also Lipschitz continuous on $\mathcal{X} \times \mathcal{Y}$.

C.2 Stochastic Approximation for Saddle Point Problems

Consider the stochastic saddle point (min-max) problem

$$\min_{x \in X} \max_{y \in Y} \Phi(x, y) = \mathbb{E}[F(x, y, \xi)]$$

where the expected value function $f(x,y)$ is convex in x and concave in y, and domains X, Y are convex closed. Let $z = [x,y]$ and $G(z,\xi) = [\nabla F_x(x,y,\xi); -\nabla F_y(x,y,\xi)]$ be the stochastic gradient for any input point z and sample ξ. Let $\|\cdot\|$ be a norm defined on the embedding Hilbert space of $Z = X \times Y$, and $D(z,z') := w(z) - w(z') - \nabla w(z')'(z-z')$ be a Bregman distance on Z defined by a 1-strongly convex (w.r.t. the norm $\|\cdot\|$) and continuously differentiable function $w(z)$. For instance, when $w(z) = \frac{1}{2}\|z\|^2$, the Bregman distance becomes $D(z,z') = \frac{1}{2}\|z-z'\|^2$.

Mirror descent SA The mirror descent stochastic approximation [70] works as follows:

$$z_i = \operatorname*{argmin}_{z\in Z}\{D(z,z_i) + \gamma_i G(z_i,\xi_i)\}, i = 1,\ldots,t.$$

The quality of an approximate solution $\bar{z} = (\bar{x},\bar{y})$ is defined by the error

$$\epsilon_{\text{gap}}(\bar{x},\bar{y}) := \max_{y\in Y}\Phi(\bar{x},y) - \Phi^* + \Phi^* - \min_{x\in X}\Phi(x,\bar{y}) = \max_{y\in Y}\Phi(\bar{x},y) - \min_{x\in X}\Phi(x,\bar{y}),$$

where Φ^* denotes the optimal value. Let $\bar{z}_t := \frac{\sum_{i=1}^t \gamma_i z_i}{\sum_{i=1}^t \gamma_i}$, the convergence properties of this weighted averaging solution is as follows.

Lemma 54 *[70] Suppose* $\mathbb{E}[\|G(z_i,\xi_i)\|_*^2] \leqslant M^2, \forall i$, *we have*

$$\mathbb{E}[\epsilon_{\text{gap}}(\bar{x}_t,\bar{y}_t)] \leqslant \frac{2\max_{z\in Z} D(z,z_1) + \frac{5}{2}M^2\sum_{i=1}^t \gamma_i^2}{\sum_{i=1}^t \gamma_i}.$$

In particular, when $\gamma_i = \frac{\gamma}{\sqrt{t}}, \forall i = 1,\ldots,t$, we have

$$\mathbb{E}[\epsilon_{\text{gap}}(\bar{x}_t,\bar{y}_t)] \leqslant (2\max_{z\in Z} D(z,z_1)/\gamma + \frac{5}{2}M^2\gamma)\frac{1}{\sqrt{t}}.$$

Moreover, suppose $D^2 = \max_{z\in Z} D(z,z_1)$ and M^2 are known, by setting $\gamma = \frac{2D}{\sqrt{5}M}$, we further have

$$\mathbb{E}[\epsilon_{\text{gap}}(\bar{x}_t,\bar{y}_t)] \leqslant \frac{2\sqrt{5}DM}{\sqrt{t}}.$$

To summarize, the mirror descent stochastic approximation achieves an $\mathcal{O}(1/\sqrt{t})$ conver-

gence rate (also known to be unimprovable [70]). Our Embedding-SGD algorithm 4 builds upon on this framework to solve the saddle point approximation problem (5.14).

C.3 Convergence Analysis for Embedding-SGD

C.3.1 Decomposition of generalization error

Let f_* be the optimal solution to our objective. Denote $\hat{L}(f) = \max_{u \in \mathcal{H}^\delta} \phi(f, u)$. Invoking the Lipschitz continuity of Φ , $L(f) - \hat{L}(f) \leqslant (K + C)\mathcal{E}(\delta), \forall f$. Therefore,

$$\begin{aligned} L(\bar{f}_t) - L(f_*) =& L(\bar{f}_t) - \hat{L}(\bar{f}_t) + \hat{L}(\bar{f}_t) - \hat{L}(f_*) + \hat{L}(f_*) - L(f_*) \\ &\leqslant \epsilon_{\text{gap}}(\bar{f}_t, \bar{u}_t) + 2(K + C)\mathcal{E}(\delta). \end{aligned}$$

C.3.2 Optimization error

Proof of Theorem 22 Our proof builds on results of stochastic approximation discussed in the previous section. Let M_1 and M_2 be such that for any $f \in \{f_i\}_{i=1}^t$ and $u \in \{u_i\}_{i=1}^t$,

$$\mathbb{E}_{x,y,z}[\|\nabla_f \hat{\Phi}_{x,y,z}(f, u)\|_{\mathcal{F}}^2] \leqslant M_1^2$$

$$\mathbb{E}_{x,y,z}[\|\nabla_u \hat{\Phi}_{x,y,z}(f, u)\|_{\mathcal{H}}^2] \leqslant M_2^2$$

Then from Lemma 54, we have

$$\mathbb{E}[\epsilon_{\text{gap}}(\bar{f}_t, \bar{y}_t)] \leqslant \frac{2(D_{\mathcal{F}}^2 + D_{\mathcal{H}}^2) + \frac{5}{2}(M_1^2 + M_2^2)\sum_{i=1}^t \gamma_i^2}{\sum_{i=1}^t \gamma_i} \tag{C.1}$$

where $D_{\mathcal{F}}^2 = \sup_{f \in \mathcal{F}} \frac{1}{2}\|f_1 - f\|_2^2$ and $D_{\mathcal{H}}^2 = \sup_{u \in \mathcal{H}^\delta} \frac{1}{2}\|u_1 - u\|_{\mathcal{H}}^2 \leqslant 2\delta$. It remains to find upper bounds for M_1 and M_2. Note that since $\|k(w, w')\|_\infty \leqslant \kappa$ for any w and w',

$$\mathbb{E}[\|\nabla_u \hat{\Phi}_{x,y,z}(f, u)\|_{\mathcal{H}}^2] \quad \leqslant \kappa \mathbb{E}[\|f(z, x) - \nabla \ell_y^*(u(x))\|^2] \leqslant 2\kappa(M_{\mathcal{F}} + c_\ell).$$

Since $u(x) = \langle u(\cdot), k(x,\cdot)\rangle_{\mathcal{H}}$, from Young's inequality, we have $|u(x)| \leqslant \frac{1}{2}\|u\|_{\mathcal{H}}^2 + \frac{1}{2}\|k(x,\cdot)\|_{\mathcal{H}}^2 \leqslant \frac{1}{2}(\delta+\kappa)$, for any $w \in \mathcal{X}$.

$$\mathbb{E}_{x,y,z}[\|\nabla_f \hat{\Phi}_{x,y,z}(f,u)\|_{\mathcal{F}}^2] \quad = \mathbb{E}[\|\psi(z,x)\|_{\mathcal{F}}^2 u(x)^2] \leqslant \tfrac{1}{4}(\delta+\kappa)^2 C_{\mathcal{F}}.$$

Plugging in $M_1^2 = 2\kappa(M_{\mathcal{F}} + c_\ell)$ and $M_2^2 = \frac{1}{4}(\delta+\kappa)^2 C_{\mathcal{F}}$ to (C.1) and setting $\gamma_t = \gamma/\sqrt{t}$, we arrive at (5.16). ■

C.4 Gradient-TD2 As Special Case of Embedding-SGD

Follow the notation in section 5.5, with the parameterization that $V^\pi(s) = \theta^T\psi(s)$ and $u(s) = \eta^T\psi(s)$, where $\psi(s) = [\psi_i(z)]_{i=1}^d \in \mathbb{R}^d$, $\theta, \eta \in \mathbb{R}^d$, the optimization becomes

$$\min_\theta \max_\eta \widehat{\Phi}(\theta,\eta) := \mathbb{E}_s \mathbb{E}_{s',a|s}\Big[\Delta_\theta(s,a,s')\psi(s)^\top \eta\Big] - \frac{1}{2}\mathbb{E}_s[\eta^\top \psi(s)\psi^\top(s)\eta], \tag{C.2}$$

where $\Delta_\theta(s,a,s') = R(s,a) + \gamma\theta^\top\psi(s') - \theta^\top\psi(s)$. For arbitrary θ, we have the closed form of $\eta(\theta)^*$ which achieves the maximum of $\widehat{\Phi}(\theta,\eta)$. Specifically, we first take derivative of $\widehat{\Phi}(\theta,\eta)$ w.r.t. η,

$$\nabla_\eta \widehat{\Phi}(\theta,\eta) = \mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')\psi(s)\right] - \mathbb{E}_s\left[\psi(s)\psi(s)^\top \eta\right], \tag{C.3}$$

and make the derivative equal to zero,

$$\eta(\theta)^* = \mathbb{E}_s\left[\psi(s)\psi(s)^\top\right]^{-1} \mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')\psi(s)\right]. \tag{C.4}$$

Plug the $\eta(\theta)^*$ into $\widehat{\Phi}(\theta,\eta)$, we achieve the optimization

$$\min_\theta \mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')\psi(s)^\top\right] \mathbb{E}_s\left[\psi(s)\psi(s)^\top\right]^{-1} \mathbb{E}_{s,a,s'}\left[\Delta_\theta(s,a,s')\psi(s)\right], \tag{C.5}$$

which is exactly the objective of gradient-TD2 [139, 140]. Plug the parametrization into the proposed embedding-SGD, we will achieve the update rules in i-th iteration proposed in

gradient-TD2 for θ and η as

$$\begin{aligned}\eta_{i+1} &= \eta_i + \gamma_i[\Delta_\theta(s,a,s') - u_i(s)]\psi(s),\\ \theta_{i+1} &= \theta_i - \gamma_i u_i(s)(\gamma\psi(s') - \psi(s)).\end{aligned}$$

Therefore, from this perspective, gradient-TD2 is simply a special case of the proposed Embedding-SGD applied to policy evalution with particular parametrization.

C.5 Dual Embedding with Arbitrary Funtion Approximator

In the main text, we only focus on using different RKHSs as the primal and dual function spaces. As we introduce in section 5.1, the proposed algorithm is versatile and can be conducted with arbitrary function space for the primal or dual functions. In this section, we demonstrate applying the algorithm to random feature represented functions [62] and neural networks. For simplicity, we specify the algorithms with either kernel, random feature representation or neural networks for both primal and dual functions. It should be emphasized that in fact the parametrization choice of the dual function is *independent* to the form of the primal function. Therefore, the algorithm can also be conducted in *hybrid setting* where the primal function uses one form of function approximator, while the dual function use another form of function approximator.

Instead of solving (5.14), in this section, we consider the alternative reformulation by penalizing the norm of the dual function, which has been widely used as an alternative to the constrained problem in machine learning literatures, and is proven to be more robust often times in practice,

$$\min_{f\in\mathcal{F}}\max_{u\in\mathcal{H}}\ \Phi(f,u) + \frac{\lambda_1}{2}\|f\|_{\mathcal{F}}^2 - \frac{\lambda_2}{2}\|u\|_{\mathcal{H}}^2 \tag{C.6}$$

It is well-known that there is a one-to-one relation between $\delta_{\mathcal{F}}$, δ and λ_1, λ_2, respectively, such that the optimal solutions to (5.14) and (C.6) are the same. The objective can also be regarded as a smoothed approximation to the original problem of our interest, see [177]. Problem (C.6) can be solved efficiently via our Algorithm 4 by simply revoking the projection operators.

C.5.1 Dual Random Feature Embeddings

In this section, we specify the proposed algorithm leveraging random feature to approximate kernel function discussed in Chapter 3. For arbitrary positive definite kernel, $k(x, x)$, there exists a measure $\rho(\omega)$ on Ω, such that $k(x, x') = \int \widehat{\phi}_w(x)\widehat{\phi}_w(x')d\rho(w)$ [20, 21], where random feature $\widehat{\phi}_w(x) : \mathcal{X} \to \mathbb{R}$ from $L_2(\mathcal{X}, \rho)$. Therefore, we can approximate the function $f \in \mathcal{H}$ with Monte-Carlo approximation $\hat{f} \in \widehat{\mathcal{H}}^m = \{\sum_{i=1}^m \beta_i \widehat{\phi}_{\omega_i}(\cdot) | \|\beta\|_2 \leqslant C\}$ where $\{w_i\}_{i=1}^m$ sampled from $\rho(\omega)$ [64]. With such an approximation, we obtain the corresponding *dual random feature embeddings* variants.

Denote the random feature for $\tilde{k}(\cdot, \cdot)$ and $k(\cdot, \cdot)$ as $\widehat{\psi}_w(\cdot)$ and $\widehat{\phi}_w(\cdot)$ with respect to distribution $\tilde{\rho}(\omega)$ and $\rho(\omega)$, respectively, we approximate the $f(\cdot)$ and $u(\cdot)$ by $\hat{f}(\cdot) = \theta^\top \widehat{\psi}(\cdot)$ and $\hat{u}(\cdot) = \eta^\top \widehat{\phi}(\cdot)$, where $\theta \in \mathbb{R}^{m\times 1}$, $\eta \in \mathbb{R}^{p\times 1}$, $\widehat{\psi}(\cdot) = [\widehat{\psi}_{\tilde{w}_1}(\cdot), \widehat{\psi}_{\tilde{w}_2}(\cdot), \ldots, \widehat{\psi}_{\tilde{w}_m}(\cdot)]^\top$ with $\{\tilde{w}_i\}_{i=1}^m \sim \tilde{\rho}(\omega)$ and $\widehat{\phi}(\cdot) = [\widehat{\phi}_{w_1}(\cdot), \widehat{\phi}_{w_2}(\cdot), \ldots, \widehat{\phi}_{w_m}(\cdot)]^\top$ with $\{w_i\}_{i=1}^p \sim \rho(\omega)$. Then, we have the saddle point reformulation of (5.1),

$$\min_{\theta} \max_{\eta} \widehat{\Phi}(\theta, \eta) := \mathbb{E}_{x,y}\mathbb{E}_{z|x}\left[\theta^\top \widehat{\psi}(z, x)\widehat{\phi}(x, y)^\top \eta - l_y^*(\eta^\top \widehat{\phi}(x, y))\right] + \frac{\lambda_1}{2}\|\theta\|^2 - \frac{\lambda_2}{2}\|\eta\|^2. \quad \text{(C.7)}$$

Apply the proposed algorithm to (C.7), we obtain the update rule in i-th iteration,

$$\begin{aligned} \theta_{i+1} &= (1 - \gamma_i\lambda_1)\theta_i - \gamma_i \hat{u}(x_i, y_i)\widehat{\psi}(z_i, x_i), \\ \eta_{i+1} &= (1 - \gamma_i\lambda_1)\eta_i + \gamma_i\big[\hat{f}_i(z_i, x_i) - \nabla \ell_{y_i}^*(\hat{u}(x_i, y_i))\big]\widehat{\phi}(x_i, y_i). \end{aligned}$$

We emphasize that with the random feature representation will introduce an extra approximation error term in the order of $\mathcal{O}(1/\sqrt{m})$. To balance the approximate error and the statistical generalization error, we must use m sufficiently large.

C.5.2 Extension to Embedding Doubly-SGD

To alleviate the approximation error introduced by random feature representation, we can further generalize the algorithmic technique about doubly stochastic gradient in Chapter 3 to the saddle point problem (C.6), which can be viewed as setting m to be infinite conceptually, therefore, eliminate the approximation error due to random feature representation.

The embedding doubly-SGD is illustrated in Algorithm 6,

Algorithm 6 Embedding-Doubly SGD for (C.6)

Input: $p(x,y)$, $p(z|x)$, $\tilde{\rho}(\omega), \rho(\omega)$, $\{\gamma_i \geqslant 0\}_{i=1}^t$

for $i = 1, \ldots, t$ **do**

  Sample $x_i, y_i \sim p(x,y)$.

  Sample $z_i \sim p(z|x_i)$.

  Sample $\omega_i \sim \rho(\omega)$ with seed i

  Sample $\widetilde{\omega}_i \sim \tilde{\rho}(\omega)$ with seed $\tilde{i}$

  Compute $f_i = \mathbf{Predict}(z_i, x_i, \{\alpha_j\}_{j=1}^i)$

  Compute $u_i = \mathbf{Predict}(x_i, y_i, \{\beta_j\}_{j=1}^i)$

  $\alpha_{i+1} = \gamma_i u_i(x_i, y_i)\widehat{\psi}_{\widetilde{\omega}_i}(z_i, x_i)$.

  $\beta_{i+1} = \gamma_i[f_i(z_i, x_i) - \nabla \ell_{y_i}^*(u_i(x_i, y_i))]\widehat{\phi}_{\omega_i}(x_i, y_i)$.

  for $j = 1, \ldots, i$

    $\alpha_j = (1 - \gamma_i \lambda_1)\alpha_j$, $\beta_j = (1 - \gamma_i \lambda_2)\beta_j$

end for

Algorithm 7 $u = \mathbf{Predict}(x, y, \{\beta_i\}_{i=1}^t)$

Input: $\rho(\omega)$, $\widehat{\phi}_\omega(x,y)$.

Set $u = 0$.

for $i = 1, \ldots, t$ **do**

  Sample $\omega_i \sim \rho(\omega)$ with seed i.

  $u = u + \beta_i \widehat{\phi}_{\omega_i}(x,y)$.

end for

Algorithm 8 $f = \textbf{Predict}(z, x, \{\alpha_i\}_{i=1}^t)$

1: **Input:** $\tilde{\rho}(\omega)$, $\widehat{\psi}_\omega(z, x)$.

2: Set $f = 0$.

3: **for** $i = 1, \ldots, t$ **do**

4: Sample $\widetilde{\omega}_i \sim \tilde{\rho}(\omega)$ with seed $\tilde{i}$.

5: $f = f + \alpha_i \widehat{\psi}_{\widetilde{\omega}_i}(z, , x)$.

6: **end for**

C.5.3 Dual Neural Networks Embeddings

To achieve better performance with fewer basis functions, we can also learn the basis functions $\widehat{\psi}(\cdot)$ and $\widehat{\phi}(\cdot)$ jointly with θ and η by back-propagation. Specifically, denote the parameters in $\widehat{\psi}(\cdot) = [\widehat{\psi}_{\tilde{w}_1}(\cdot), \widehat{\psi}_{\tilde{w}_2}(\cdot), \ldots, \widehat{\psi}_{\tilde{w}_m}(\cdot)]^\top$ and $\widehat{\phi}(\cdot) = [\widehat{\phi}_{w_1}(\cdot), \widehat{\phi}_{w_2}(\cdot), \ldots, \widehat{\phi}_{w_m}(\cdot)]^\top$ explicitly as $\widetilde{W} = [\tilde{w}_i]_{i=1}^m$ and $W = [w_i]_{i=1}^m$, we also include $\tilde{W}$ and W into optimization (C.7), which results

$$\begin{aligned}\min_{\theta, \widetilde{W}} \max_{\eta, W} \widehat{\Phi}(\theta, \widetilde{W}, \eta, W) \quad &:= \quad \mathbb{E}_{x,y}\mathbb{E}_{z|x}\left[\theta^\top \widehat{\psi}_{\widetilde{W}}(z, x)\widehat{\phi}_W(x, y)^\top \eta - l_y^*\left(\eta^\top \widehat{\phi}_W(x, y)\right)\right] \\ &\quad + \frac{\lambda_1}{2}\|\theta\|^2 - \frac{\lambda_2}{2}\|\eta\|^2. \end{aligned} \tag{C.8}$$

Apply the proposed algorithm to (C.8), we obtain the update rule for all the parameters, $\{\theta, \eta, \widetilde{W}, W\}$, in i-th iteration,

$$\begin{aligned}
\theta_{i+1} &= (1 - \gamma_i\lambda_1)\theta_i - \gamma_i \eta_i^\top \widehat{\phi}_{W_i}(x_i, y_i)\widehat{\psi}_{\widetilde{W}_i}(z_i, x_i), \\
\eta_{i+1} &= (1 - \gamma_i\lambda_1)\eta_i + \gamma_i\left[\theta_i^\top \widehat{\psi}_{\widetilde{W}_i}(z_i, x_i) - \nabla \ell_{y_i}^*\left(\eta_i^\top \widehat{\phi}_{W_i}(x_i, y_i)\right)\right]\widehat{\phi}_{W_i}(x_i, y_i), \\
\widetilde{W}_{i+1} &= \widetilde{W}_i - \gamma_i \eta_i^\top \widehat{\phi}_{W_i}(x_i, y_i)\theta_i^\top \nabla_{\widetilde{W}} \widehat{\psi}_{\widetilde{W}}(z_i, x_i), \\
W_{i+1} &= W_i + \gamma_i\left[\theta_i^\top \widehat{\psi}_{\widetilde{W}_i}(z_i, x_i) - \nabla \ell_{y_i}^*\left(\eta_i^\top \widehat{\phi}_{W_i}(x_i, y_i)\right)\right]\eta_i^\top \nabla_W \widehat{\phi}_W(x_i, y_i).
\end{aligned}$$

Here we only demonstrate the back-propagation algorithm applies to one-layer basis functions, in fact, it can be extended to the deep basis functions, *i.e.*, hierarchical composition functions, straight-forwardly if necessary. With such deep neural networks as function approximator in our algorithm, we achieve the dual neural networks embeddings.

C.6 Actor-Critic Algorithm with Dual Embedding

We can extend the same technique to control problem. Different from the policy evaluation in which the policy is provided, the control problem is trying to learn the optimal policy in terms of reward, *i.e.*, the policy which can achieve the maximum reward.

We define the action-value function $Q^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which evaluates the value of taking action a in state s with policy π for future actions,

$$Q^\pi(s,a) = \mathbb{E}\bigg[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}) \bigg| s_0 = s, A_0 = a, \pi\bigg].$$

Based on the definition of the Q^π function, we have the Bellman equation as

$$Q^\pi(s,a) = R(s) + \gamma \mathbb{E}_{s' \sim P(s'|s,a), a' \sim \pi(a|s')}[Q^\pi(s',a')|s,a], \tag{C.9}$$

therefore, we obtain the optimization as

$$\min_Q L(Q) := \mathbb{E}_{s\sim\mu(s), a\sim\pi(a|s)}\bigg[\bigg(\mathbb{E}_{s',a'|s}[R(s) + \gamma Q(s',a')] - Q(s,a)\bigg)^2\bigg] \tag{C.10}$$

whose objective is the same as SARSA. Apply the dual kernel embedding, we obtain

$$\min_Q \max_{u\in\mathcal{H}} \mathbb{E}_{s,a}\bigg[\bigg\langle \mathbb{E}_{s',a'|s}[R(s) + \gamma Q(s',a')] - Q(s,a), u(s,a)\bigg\rangle\bigg] - \frac{1}{2}\mathbb{E}_{s,a}[u(s,a)^2], \tag{C.11}$$

which can be solved efficiently. Recall the definition of Q function, our framework can be also extended to λ-return for the estimation, *i.e.*, $G(\lambda) = (1-\lambda)\sum_{t=1}^{\infty} \lambda^{t-1} G^{(n)}$ where $G^{(n)} = \sum_{t=0}^{n} \gamma^t R(s_t)$.

With the proposed Q–function learning procedure, we can achieve the control target by *policy iteration* in Algorithm 9. The convergence of this procedure can be proved by taking the approximate error from Q function into account in the policy iteration convergence rate. [221, 222] already provides a framework for the analysis and we omit the details here.

Algorithm 9 Policy Iteration with Q-Dual Kernel Embedding

Initialize $Q(s,a)$ randomly
for $i = 1, \ldots, t$ **do**
  update $\pi_i(a|s)$ as ϵ-greedy policy from $Q_i(s,a)$
  for $j = 1, \ldots, n$ **do**
    sample $s \sim \mu(s)$, $a \sim \pi_i(a|s)$, $s' \sim p(s'|s,a)$ and $a' \sim \pi_i(a|s')$
    update Q_i^j by applying Algorithm 4 to (C.11)
  end for
  $Q_{i+1} = Q_i^n$
end for

C.7 Policy Evaluation with Dynamics Kernel and Random Feature Conditional Embeddings

In the policy evaluation experiment section, we compared the proposed algorithm with the MDPs embedding using kernels. [153] introduces the RKHS embedding of transition dynamics in MDPs utilizing kernel conditional embedding [137]. With the nonparametric representation of the transition, the expectation can be computed easily and accurately by linear operatior, and thus, promoting the performance of policy evaluation and control task. For self-containing, we specify the details of MDPs embeddings with kernel and the random feature extension in this section. Moreover, to utilize data sequantially, we propose the algorithm for policy evaluation utilizing (functional) stochastic gradient with both kernel and random feature conditional embeddings, which is different from the value iteration algorithm with kernel embedded MDP in [153].

C.7.1 Dynamics RKHS Embeddings

In MDPs, the transition dynamics, $p(s'|a,s)$, plays a vital. Many methods for solving MDPs requires the computation with respect to the dynamics, *i.e.*, $\mathbb{E}_{s' \sim p(s'|a,s)}[f(s')]$. By the kernel conditional embedding estimation [137], the expectation can be estimated from data. Specifically, given data $\{(s_i, a_i, s_i')\}_{i=1}^m$, kernel $k : (\mathcal{X} \times \mathcal{A}) \times (\mathcal{X} \times \mathcal{A}) \to \mathbb{R}$ and $\tilde{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the expectation for $\forall f \in \tilde{\mathcal{H}}$ can be approximated by $\langle \tau((s,a),\cdot), f \rangle$ with $\tau((s,a),\cdot) \in \tilde{\mathcal{H}}$ as

$$\tau_{(s,a)} = \alpha(s,a)^\top \widetilde{K}(S', \cdot), \tag{C.12}$$

where $\widetilde{K}(S', \cdot) = [\tilde{k}(s'_1, \cdot), \tilde{k}(s'_2, \cdot), \ldots, \tilde{k}(s'_m, \cdot)]^\top$, $\alpha(s, a) = (K + \lambda m I)^{-1} K((S, A), (s, a))$, $K((S, A), (s, a)) = [k((s_1, a_1), (s, a)), k((s_2, a_2), (s, a)), \ldots, k((s_m, a_m), (s, a))]^\top$ and $K = [k((s_i, a_i), (s_j, a_j))]_{i,j=1}^m$, λ is a regularization parameter.

With the RKHS embedded transition dynamics, we can compute the expectation

$$\mathbb{E}_{s' \sim p(s'|a,s)}[f(s')] \approx \langle \tau_{(s,a)}, f \rangle.$$

The theoretical property of the estimator is analyzed in [137, 138].

C.7.2 Dynamics Random Feature Embeddings

We can also leverage the random feature to approximate the kernel function in kernel conditional embedding, which will result the random feature embedding for memory efficiency. With the same notations, we denote the random features for $\tilde{k}$ and k are $\widehat{\psi}_w(\cdot)$ and $\widehat{\phi}_w(\cdot)$ with respect to $\widetilde{\rho}(\omega)$ and $\rho(\omega)$ respectively, we can approximate the kernel conditional embedding (C.12) by

$$\widehat{\tau}_{(s,a)} = \widehat{\phi}(s, a)^\top (\widehat{\Phi}\widehat{\Phi}^\top + \lambda m I)^{-1} \widehat{\Phi}\widehat{\Psi}^\top, \tag{C.13}$$

where

$$\widehat{\phi}(s, a) = [\widehat{\phi}_{w_1}(s, a), \widehat{\phi}_{w_2}(s, a), \ldots, \widehat{\phi}_{w_p}(s, a)]^\top \in \mathbb{R}^{p \times 1},$$

$$\widehat{\psi}(s) = [\widehat{\psi}_{\tilde{w}_1}(s), \widehat{\psi}_{\tilde{w}_2}(s), \ldots, \widehat{\psi}_{\tilde{w}_p}(s)] \in \mathbb{R}^{p \times 1},$$

$$\widehat{\Phi} = [\widehat{\phi}(s_1, a_1), \widehat{\phi}(s_2, a_2), \ldots, \widehat{\phi}(s_m, a_m)] \in \mathbb{R}^{p \times m}$$

and

$$\widehat{\Psi} = [\widehat{\psi}(s'_1), \widehat{\psi}(s'_2), \ldots, \widehat{\psi}(s'_m)] \in \mathbb{R}^{p \times m}.$$

Then, with random feature represented $f(s') = \theta^\top \widehat{\psi}(s')$, the expectation $\mathbb{E}_{s' \sim p(s'|a,s)}[f(s')]$ can be approximated by $\widehat{\tau}_{(s,a)}\theta$.

C.7.3 Policy Evaluation with Dynamics Embeddings

As we introduced, with the kernel and random feature embedded transition dynamics, the expectation can be estimated with linear operator. Instead of plugging the embedded MDP into value iteration for policy evaluation in [153], which is not suitable for dynamics random feature embeddings, we propose new algorithms by applying the (functional) stochastic gradient algorithm to optimize mean-square Bellman error with respect to $V(\cdot)$ in kernel representation or θ in random feature representation, we have

$$\nabla_{V(\cdot)} L = \varpi(a|s)(V(s) - (R(s,a) + \gamma\langle\tau(s,a), V\rangle))(\tilde{k}(s,\cdot) - \gamma\mathbb{E}_{s'|a,s}[\tilde{k}(s',\cdot)]) \quad \text{(C.14)}$$

$$\nabla_{\theta} L = \varpi(a|s)(\theta^\top\widehat{\psi}(s) - (R(s,a) + \gamma\widehat{\tau}(s,a)\theta))(\widehat{\psi}(s) - \gamma\widehat{\tau}(s,a)^\top) \quad \text{(C.15)}$$

where $\mathbb{E}_{s'|a,s}[\tilde{k}(s',\cdot)]$ can be approximated by $\tau(a,s)$ [137]. Therefore, plugging the gradient estimator into SGD results the following two algorithms for kernel/random feature embedded dynamics.

Algorithm 10 SGD with MDP kernel embedding

Input: $\pi(\cdot), \mu$.

1: compute $\tau = (K + \lambda m I)^{-1}$

2: **for** $i = 1, \ldots, t$ **do**

3: Sample $s \sim \mu(s), a \sim \pi_b(a|s), s' \sim p(s'|s,a)$.

4: $\bar{v} = K((S,A),(s,a))^\top \tau V_i(S')$.

5: $\bar{k}(\cdot) = K((S,A),(s,a))^\top \tau \widetilde{K}(S',\cdot)$

6: $V_{i+1} = (1 - \gamma_i\lambda_1)V_i - \gamma_i\varpi(a|s)(V(s') - (R(s) + \gamma\bar{v})(\tilde{k}(s,\cdot) - \gamma\bar{k}(\cdot))$.

7: **end for**

Algorithm 11 SGD with MDP random feature embedding

Input: $\pi(\cdot), \mu, \widehat{\phi}(\cdot), \widehat{\psi}(\cdot)$.

compute $\tau = \left(\widehat{\Phi}\widehat{\Phi}^\top + \lambda m I\right)^{-1} \widehat{\Phi}\widehat{\Psi}^\top$

for $i = 1, \ldots, t$ **do**

  Sample $s \sim \mu(s), a \sim \pi_b(a|s), s' \sim p(s'|s, a)$.

  $\theta_{i+1} = (1 - \gamma_i \lambda_1)\theta_i - \gamma_i \varpi(a|s)(\theta_i^\top \widehat{\psi}(s') - (R(s) + \gamma \widehat{\phi}(s, a)^\top \tau\theta))(\widehat{\psi}(s') - \gamma \widehat{\phi}(s, a)^\top \tau)$.

end for

APPENDIX D
PROOF DETAILS AND EXTENSION OF SBEED IN CHAPTER 6

D.1 Properties of Smoothed Bellman Operator

After applying the smoothing technique [177], we obtain a new Bellman operator, $\widetilde{\mathcal{T}}$, which is contractive. By such a property, we can guarantee the uniqueness of the solution; a similar result is also presented in [180, 183].

Proposition 24 (Contraction) *$\mathcal{T}_\lambda$ is a γ-contraction. Consequently, the corresponding smoothed Bellman equation (6.4), or equivalently (6.5), has a unique solution V_λ^*.*

Proof For any $V_1, V_2 : \mathcal{S} \to \mathbb{R}$, we have

$$
\begin{aligned}
& \left\| \widetilde{\mathcal{T}} V_1 - \widetilde{\mathcal{T}} V_2 \right\|_\infty \\
= \; & \left\| \max_\pi \left\{ \left\langle \pi, R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V_1(s') \right] \right\rangle + \lambda H(\pi) \right\} - \max_\pi \left\{ \left\langle \pi, R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V_2(s') \right] \right\rangle + \lambda H(\pi) \right\} \right\|_\infty \\
\leqslant \; & \left\| \max_\pi \left\{ \left\langle \pi, R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V_1(s') \right] \right\rangle + \lambda H(\pi) - \left\langle \pi, R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V_2(s') \right] \right\rangle - \lambda H(\pi) \right\} \right\| \\
\leqslant \; & \left\| \max_\pi \left\langle \pi, \gamma \mathbb{E}_{s'|s,a} \left[V_1(s') - V_2(s') \right] \right\rangle \right\|_\infty \\
\leqslant \; & \gamma \left\| V_1 - V_2 \right\|_\infty .
\end{aligned}
$$

$\mathcal{T}_\lambda$ is therefore a γ-contraction and, by the Banach fixed point theorem, admits a unique fixed point. ■

Moreover, we may characterize the bias introduced by the entropic smoothing, similar to the simulation lemma (see, e.g., [223] and [224]):

Proposition 25 (Smoothing bias) *Let V^* and V_λ^* be the fixed points of (6.2) and (6.4), respectively. It holds that*

$$\left\| V^* - V_\lambda^* \right\|_\infty \leqslant \frac{\lambda H^*}{1-\gamma}.$$

As $\lambda \to 0$, V_λ^* converges to V^* pointwisely.

Proof Using the triangle inequality and the contraction property of $\mathcal{T}_\lambda$, we have

$$\begin{aligned}
\|V^* - V_\lambda^*\|_\infty &= \|\mathcal{T}_B V^* - \mathcal{T}_\lambda V_\lambda^*\|_\infty \\
&= \|V^* - \mathcal{T}_\lambda V^* + \mathcal{T}_\lambda V^* - \mathcal{T}_\lambda V_\lambda^*\|_\infty \\
&\leqslant \|V^* - \mathcal{T}_\lambda V^*\|_\infty + \|\mathcal{T}_\lambda V^* - \mathcal{T}_\lambda V_\lambda^*\|_\infty \\
&\leqslant \lambda H^* + \gamma \|V^* - V_\lambda^*\|_\infty ,
\end{aligned}$$

which immediately implies the desired bound. ■

The smoothed Bellman equation involves a log-sum-exp operator to approximate the max-operator, which increases the nonlinearity of the equation. We further characterize the solution of the smoothed Bellman equation, by the temporal consistency conditions.

Theorem 26 (Temporal consistency) *Assume $\lambda > 0$. Let V_λ^* be the fixed point of (6.4) and π_λ^* the corresponding policy that attains the maximum on the RHS of (6.4). Then, $(V, \pi) = (V_\lambda^*, \pi_\lambda^*)$ if and only if (V, π) satisfies the following equality for all $(s, a) \in \mathcal{S} \times \mathcal{A}$:*

$$V(s) = R(s, a) + \gamma \mathbb{E}_{s'|s,a} \left[V(s') \right] - \lambda \log \pi(a|s) . \tag{7}$$

Proof The proof has two parts.

(Necessity) We need to show $(V_\lambda^*, \pi_\lambda^*)$ is a solution to (6.6). Simple calculations give the closed form of π_λ^*:

$$\pi_\lambda^*(a|s) = Z(s)^{-1} \exp\left(\frac{R(s, a) + \gamma \mathbb{E}_{s'|s,a} \left[V_\lambda^*(s') \right]}{\lambda} \right) ,$$

where $Z(s) := \sum_{a \in \mathcal{A}} \exp\left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a} \left[V_\lambda^*(s') \right]}{\lambda} \right)$ is a state-dependent normalization con-

stant. Therefore, for any $a \in \mathcal{A}$,

$$\begin{aligned} & R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V_\lambda^*(s')\right] - \lambda \log \pi_\lambda^*(a|s) \\ = \;& R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V_\lambda^*(s')\right] - \lambda \left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[V_\lambda^*(s')\right]}{\lambda} - \log Z(s) \right) \\ = \;& \lambda \log Z(s) \;=\; V_\lambda^*(s)\,, \end{aligned}$$

where the last step is from (6.5). Therefore, $(V_\lambda^*, \pi_\lambda^*)$ satisfies (6.6).

(Sufficiency) Assume $\bar{V}$ and $\bar{\pi}$ satisfies (6.6), then we have for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ that

$$\begin{aligned} \bar{V}(s) &= R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[\bar{V}(s')\right] - \lambda \log \bar{\pi}(a|s) \\ \pi(a|s) &= \exp\left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[\bar{V}(s')\right] - \bar{V}(s)}{\lambda} \right). \end{aligned}$$

Recall $\pi(\cdot|s) \in \mathcal{P}$, we have

$$\begin{aligned} & \sum_{a \in \mathcal{A}} \exp\left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[\bar{V}(s')\right] - \bar{V}(s)}{\lambda} \right) = 1 \\ \Rightarrow \quad & \sum_{a \in \mathcal{A}} \exp\left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[\bar{V}(s')\right]}{\lambda} \right) = \exp\left(\frac{\bar{V}(s)}{\lambda} \right) \\ \Rightarrow \quad & \bar{V}(s) = \lambda \log \left(\sum_{a \in \mathcal{A}} \exp\left(\frac{R(s,a) + \gamma \mathbb{E}_{s'|s,a}\left[\bar{V}(s')\right]}{\lambda} \right) \right) = \mathcal{T}_\lambda \bar{V}(s)\,. \end{aligned}$$

The last equation holds for all $s \in \mathcal{S}$, so $\bar{V}$ is a fixed point of $\widetilde{\mathcal{T}}$. It then follows from Proposition 24 that $\bar{V} = V_\lambda^*$. Finally, $\bar{\pi} = \pi_\lambda^*$ due to strong concavity of the entropy function ■

The same conditions have been re-discovered several times, *e.g.*, [179, 182], from a completely different point of views.

D.2 Variance Cancellation via the Min-Max Formulation

The second term in the min-max formulation (6.9) will cancel the variance $\mathbb{V}_{s,a,s'}\left[\gamma V(s')\right]$. Formally,

Proposition 55 *Given any fixed (V, π), we have*

$$\max_{\varsigma \in \mathcal{F}(\mathcal{S}\times\mathcal{A})} -\mathbb{E}_{s,a,s'}\left[\big(R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)-\varsigma(s,a)\big)^2\right] = -\gamma^2\mathbb{E}_{s,a}\left[\mathbb{V}_{s'|s,a}\left[V(s')\right]\right] \quad \text{(D.1)}$$

Proof Recall from (6.9) that $\delta(s,a,s') = R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)$. Then,

$$\begin{aligned} & \max_{\varsigma} \ -\mathbb{E}_{s,a,s'}\left[\big(R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)-\varsigma(s,a)\big)^2\right] \\ = \ & -\min_{\varsigma} \ \mathbb{E}_{s,a}\left[\mathbb{E}_{s'|s,a}\left[\big(\delta(s,a,s')-\varsigma(s,a)\big)^2\right]\right] . \end{aligned}$$

Clearly, the minimizing function ς^* may be determined for each (s,a) entry separately. Fix any $(s,a) \in \mathcal{S}\times\mathcal{A}$, and define a function on $\mathbb{R}$ as $q(x) := \mathbb{E}_{s'|s,a}\left[(\delta(s,a,s')-x)^2\right]$. Obviously, this convex function is minimized at the stationary point $x^* = \mathbb{E}_{s'|s,a}\left[\delta(s,a,s')\right]$. We therefore have $\varsigma^*(s,a) = \mathbb{E}_{s'|s,a}\left[\delta(s,a,s')\right]$ for all (s,a), so

$$\begin{aligned} & \min_{\varsigma} \ \mathbb{E}_{s,a}\left[\mathbb{E}_{s'|s,a}\left[\big(\delta(s,a,s')-\varsigma(s,a)\big)^2\right]\right] \\ = \ & \mathbb{E}_{s,a}\left[\mathbb{E}_{s'|s,a}\left[\big(\delta(s,a,s')-\mathbb{E}_{s'|s,a}\left[\delta(s,a,s')\right]\big)^2\right]\right] \\ = \ & \mathbb{E}_{s,a}\left[\mathbb{V}_{s'|s,a}\left[\delta(s,a,s')\right]\right] \\ = \ & \mathbb{E}_{s,a}\left[\mathbb{V}_{s'|s,a}\left[\gamma V(s')\right]\right] \ = \ \gamma^2\mathbb{E}_{s,a}\left[\mathbb{V}_{s'|s,a}\left[V(s')\right]\right] , \end{aligned}$$

where the second last step is due to the fact that, conditioned on s and a, the only random variable in $\delta(s,a,s')$ is $V(s')$. ■

D.3 Details of SBEED

In this section, we provide further details of the SBEED algorithms, including its gradient derivation and multi-step/eligibility-trace extension.

D.3.1 Unbiasedness of Gradient Estimator

In this subsection, we compute the gradient with respect to the primal variables. Let (w_V, w_π) be the parameters of the primal (V,π), and w_ς the parameters of the dual ς. Abus-

ing notation a little bit, we now write the objective function $L_\eta(V, \pi; \varsigma)$ as $L_\eta(w_V, w_\pi; w_\varsigma)$. Recall the quantity $\delta(s, a, s')$ from (6.9).

Theorem 27 (Gradient derivation) *Define* $\bar{\ell}_\eta(w_V, w_\pi) := L_\eta(w_V, w_\pi; w_\varsigma^*)$*, where* $w_\varsigma^* = \arg\max_{w_\varsigma} L_\eta(w_V, w_\pi; w_\varsigma)$*. Let* $\delta_{s,a,s'}$ *be a shorthand for* $\delta(s, a, s')$*, and* $\hat{\varsigma}$ *be dual parameterized by* w_ς^**. Then,*

$$\begin{aligned}
\nabla_{w_V} \bar{\ell}_\eta =& 2\mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - V(s)\right)\left(\gamma \nabla_{w_V} V(s') - \nabla_{w_V} V(s)\right)\right] \\
& - 2\eta\gamma \mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - \hat{\varsigma}(s,a)\right)\nabla_{w_V} V(s')\right], \\
\nabla_{w_\pi} \bar{\ell}_\eta =& -2\lambda \mathbb{E}_{s,a,s'}\left[\left((1-\eta)\delta_{s,a,s'} + \eta\hat{\varsigma}(s,a) - V(s)\right)\cdot \nabla_{w_\pi} \log \pi(a|s)\right].
\end{aligned}$$

Proof First, note that w_ς^* is an implicit function of (w_V, w_π). Therefore, we must use the chain rule to compute the gradient:

$$\begin{aligned}
\nabla_{w_V} \bar{\ell}_\eta \;=\; & 2\mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - V(s; w_V)\right)\left(\gamma \nabla_{w_V} V(s'; w_V) - \nabla_{w_V} V(s; w_V)\right)\right] \\
& -2\eta\gamma \mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - \varsigma(s,a; w_\varsigma^*)\right)\nabla_{w_V} V(s'; w_V)\right] \\
& +2\eta\gamma \mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - \varsigma(s,a; w_\varsigma^*)\right)\nabla_{w_V} \varsigma(s,a; w_\varsigma^*)\right].
\end{aligned}$$

We next show that the last term is zero:

$$\begin{aligned}
& \mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - \varsigma(s,a; w_\varsigma^*)\right)\nabla_{w_V} \varsigma(s,a; w_\varsigma^*)\right] \\
=\; & \mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - \varsigma(s,a; w_\varsigma^*)\right)\cdot \nabla_{w_V} w_\varsigma^* \cdot \nabla_{w_\varsigma} \varsigma(s,a; w_\varsigma^*)\right] \\
=\; & \nabla_{w_V} w_\varsigma^* \cdot \mathbb{E}_{s,a,s'}\left[\left(\delta_{s,a,s'} - \varsigma(s,a; w_\varsigma^*)\right)\cdot \nabla_{w_\varsigma} \varsigma(s,a; w_\varsigma^*)\right] \\
=\; & \nabla_{w_V} w_\varsigma^* \cdot \mathbf{0} \;=\; \mathbf{0},
\end{aligned}$$

where the first step is the chain rule; the second is due to the fact that $\nabla_{w_V} w_\varsigma^*$ is not a function of (s, a, s'), so can be moved outside of the expectation; the third step is due to the optimality of w_ς^*. The gradient w.r.t. w_V is thus derived. The case for w_π is similar. ■

D.3.2 Multi-step Extension

One way to interpret the smoothed Bellman equation (6.4) is to treat each $\pi(\cdot|s)$ as a (mixture) action; in other words, the action space is now the simplex $\mathcal{P}_{\mathcal{A}}$. With this interpretation, the introduced entropy regularization may be viewed as a shaping reward: given a mixture action $\pi(\cdot|s)$, its immediate reward is given by

$$\tilde{R}(s, \pi(\cdot|s)) := \mathbb{E}_{a\sim\pi(\cdot|s)}\left[R(s,a)\right] + \lambda H(\pi, s)\,.$$

The transition probabilities can also be adapted accordingly as follows

$$\tilde{P}(s'|s, \pi(\cdot|s)) := \mathbb{E}_{a\in\pi(\cdot|s)}\left[P(s'|s,a)\right]\,.$$

It can be verified that the above constructions induce a well-defined MDP $\tilde{M} = \langle \mathcal{S}, \mathcal{P}_{\mathcal{A}}, \tilde{P}, \tilde{R}, \gamma\rangle$, whose standard Bellman equation is exactly (6.4).

With this interpretation, the proposed framework and algorithm can be easily applied to multi-step and eligibility-traces extensions. Specifically, one can show that $(V^*_\lambda, \pi^*_\lambda)$ is the unique solution that satisfies the multi-step expansion of (6.6): for any $k \geqslant 1$ and any $(s_0, a_0, a_1, \ldots, a_{k-1}) \in \mathcal{S}\times\mathcal{A}^k$,

$$V(s_0) = \sum_{t=0}^{k-1} \gamma^t \mathbb{E}_{s_t|s_0,a_{0:t-1}}\left[R(s_t,a_t) - \lambda\log\pi(a_t|s_t)\right] + \gamma^k \mathbb{E}_{s_k|s_0,a_{0:k-1}}\left[V(s_k)\right]\,. \tag{D.2}$$

Clearly, when $k = 1$ (the single-step bootstrapping case), the above equation reduces to (6.6).

The k-step extension of objective function (6.7) now becomes

$$\min_{V,\pi} \mathbb{E}_{s_0,a_{0:k-1}}\left[\left(\sum_{t=0}^{k-1}\gamma^t\mathbb{E}_{s_t|s_0,a_{0:t-1}}\left[R(s_t,a_t) - \lambda\log\pi(a_t|s_t)\right] + \gamma^k\mathbb{E}_{s_k|s_0,a_{0:k-1}}\left[V(s_k)\right] - V(s_0)\right)^2\right].$$

Applying the Legendre-Fenchel transformation and the interchangeability principle, we

arrive at the following multi-step primal-dual optimization problem:

$$\begin{aligned}\min_{V,\pi}\max_{\nu} \quad & \mathbb{E}_{s_0,a_{0:t-1}}\Bigg[\nu(s_0,a_{0:t-1})\Bigg(\sum_{t=0}^{k-1}\gamma^t\mathbb{E}_{s_t|s_0,a_{0:k-1}}\left[R(s_t,a_t)-\lambda\log\pi(a_t|s_t)\right] \\ & +\gamma^k\mathbb{E}_{s_k|s_0,a_{0:k-1}}\left[V(s_k)\right]-V(s_0)\Bigg)\Bigg]-\frac{1}{2}\mathbb{E}_{s_0,a_{0:k-1}}\left[\nu(s_0,a_{0:k-1})^2\right] \\ =\min_{V,\pi}\max_{\nu} \quad & \mathbb{E}_{s_{0:k},a_{0:k-1}}\Bigg[\nu(s_0,a_{0:k-1})\Bigg(\sum_{t=0}^{k-1}\gamma^t\left(R(s_t,a_t)-\lambda\log\pi(a_t|s_t)\right) \\ & +\gamma^kV(s_k)-V(s_0)\Bigg)\Bigg]-\frac{1}{2}\mathbb{E}_{s_0,a_{0:k-1}}\left[\nu(s_0,a_{0:k-1})^2\right].\end{aligned}$$

Similar to the single-step case, defining

$$\delta(s_{0:k},a_{0:k-1}):=\sum_{t=0}^{k-1}\gamma^t\left(R(s_t,a_t)-\lambda\log\pi(a_t|s_t)\right)+\gamma^kV(s_k).$$

and using the substitution $\varsigma(s_0,a_{0:k-1})=\nu(s_0,a_{0:k-1})+V(s_0)$, we reach the following min-max formulation:

$$\min_{V,\pi}\max_{\varsigma}L(V,\pi;\varsigma):=\mathbb{E}_{s_{0:k},a_{0:k-1}}\left[\left(\delta(s_{0:k},a_{0:k-1})-V(s_0)\right)^2-\eta\left(\delta(s_{0:k},a_{0:k-1})-\varsigma(s_0,a_{0:k-1})\right)^2\right] \tag{D.3}$$

where the dual function now is $\varsigma(s_0,a_{0:k-1})$, a function on $\mathcal{S}\times\mathcal{A}^k$, and $\eta\geqslant 0$ is again a parameter used to balance between bias and variance. It is straightforward to generalize Theorem 27 to the multi-step setting, and to adapt SBEED accordingly,

D.3.3 Eligibility-trace Extension

Eligibility traces can be viewed as an approach to aggregating multi-step bootstraps for $k\in\{1,2,\cdots\}$; see [2] for more discussions. The same can be applied to the multi-step consistency condition (D.2), using an exponential weighting parameterized by $\zeta\in[0,1)$. Specifically, for all $(s_0,a_{0:k-1})\in\mathcal{S}\times\mathcal{A}^k$, we have

$$V(s_0)=(1-\zeta)\sum_{k=1}^{\infty}\zeta^{k-1}\left(\sum_{t=0}^{k-1}\gamma^t\mathbb{E}_{s_t|s_0,a_{0:k-1}}\left[R(s_t,a_t)-\lambda\log\pi(a_t|s_t)\right]+\gamma^k\mathbb{E}_{s_k|s_0,a_{0:k-1}}\left[V(s_k)\right]\right) \tag{D.4}$$

Then, following similar steps as in the previous subsection, we reach the following min-max optimization:

$$\min_{V,\pi}\max_{\varsigma} \quad \mathbb{E}_{s_{0:\infty},a_{0:\infty}}\left[\left((1-\zeta)\sum_{k=1}^{\infty}\zeta^{k-1}\delta(s_{0:k},a_{0:k-1})-V(s_0)\right)^2\right] \\ -\eta\,\mathbb{E}_{s_{0:\infty},a_{0:\infty}}\left[\left((1-\zeta)\sum_{k=1}^{\infty}\zeta^{k-1}\delta(s_{0:k},a_{0:k-1})-\varsigma(s_0,a_{0:\infty})\right)^2\right]. \quad \text{(D.5)}$$

In practice, $\varsigma(s_0, a_{0:\infty})$ can be parametrized by neural networks with finite length of actions as input as an approximation.

D.4 Proof Details of the Theoretical Analysis

In this section, we provide the details of the analysis in Theorems 29 and 30. We start with the boundedness of V^* and V_λ^* under Assumption 10. Given any measure on the state space $\mathcal{S}$,

$$\|V^*\|_\mu \leqslant \|V^*\|_\infty \leqslant (1+\gamma+\gamma^2+\cdots)C_R = C_V := \frac{C_R}{1-\gamma}.$$

A similar argument may be used on V_λ^* to get

$$\|V_\lambda^*\|_\mu \leqslant \frac{C_R+H^*}{1-\gamma}.$$

It should be emphasized that although Assumption 10 ensures boundedness of V^* and $\log\pi^*(a|s)$, it does not imply the continuity and smoothness. In fact, as we will see later, λ controls the trade-off between approximation error (due to parameterization) and bias (due to smoothing) in the solution of the smoothed Bellman equation.

D.4.1 Error Decomposition

Recall that

- (V^*, π^*) corresponds to the optimal value function and optimal policy to the original Bellman equation, namely, they are solutions to the optimization problem (6.3);
- $(V_\lambda^*, \pi_\lambda^*)$ corresponds to the optimal value function and optimal policy to the smoothed

Bellman equation, namely, they are solutions to the optimization problem (6.7) with objective $\ell(V, \pi)$;

- (V_w^*, π_w^*) correponds to the optimal solution to the optimization problem (6.7) under nonlinear function approximation, with objective $\ell_w(V_w, \pi_w)$;
- $(\widehat{V}_w^*, \widehat{\pi}_w^*)$ stands for the optimal solution to the finite sample approximation of (6.7) under nonlinear function approximation, with objective $\widehat{\ell}_T(V_w, \pi_w)$.

Hence, we can decompose the error between $(\widehat{V}_w^*, \widehat{\pi}_w^*)$ and $(V_\lambda^*, \pi_\lambda^*)$ under the $\|\cdot\|_{\mu\pi_b}$ norm.

$$\left\|\widehat{V}_w^* - V^*\right\|_{\mu\pi_b}^2 \leqslant 2\left\|\widehat{V}_w^* - V_\lambda^*\right\|_{\mu\pi_b}^2 + 2\left\|V_\lambda^* - V^*\right\|_{\mu\pi_b}^2. \tag{D.6}$$

We first look at the second term from smoothing error, which can be similarly bounded, as shown in Proposition 25.

Lemma 56 (Smoothing bias) $\|V_\lambda^* - V^*\|_{\mu\pi_b}^2 \leqslant (2\gamma^2 + 2)\left(\frac{\gamma\lambda}{1-\gamma}\max_{\pi\in\mathcal{P}} H(\pi)\right)^2$.

Proof For $\|V_\lambda^* - V^*\|_{\mu\pi_b}^2$, we have

$$\begin{aligned}
\|V_\lambda^* - V^*\|_{\mu\pi_b}^2 &= \int \left(\gamma\mathbb{E}_{s'|s,a}\left[V^*(s') - V_\lambda^*(s')\right] - (V^*(s) - V_\lambda^*(s))\right)^2 \mu(s)\pi_b(a|s)dsda \\
&\leqslant 2\gamma^2\left\|\mathbb{E}_{s'|s,a}\left[V^*(s') - V_\lambda^*(s')\right]\right\|_\infty^2 + 2\left\|V^*(s) - V_\lambda^*(s)\right\|_\infty \\
&\leqslant (2\gamma^2 + 2)\left(\frac{\gamma\lambda}{1-\gamma}\max_{\pi\in\mathcal{P}} H(\pi)\right)^2,
\end{aligned}$$

where the final inequality is because Lemma 25. ■

We now look at the first term and show that

Lemma 57

$$\begin{aligned}
\left\|\widehat{V}_w^* - V_\lambda^*\right\|_{\mu\pi_b}^2 \leqslant\ & 2\left(\ell(\widehat{V}_w^*, \widehat{\pi}_w^*) - \ell(V_\lambda^*, \pi_\lambda^*)\right) + 4\lambda^2\left\|\log\widehat{\pi}_w^*(a|s) - \log\hat{\pi}_w^*(a|s)\right\|_2^2 \\
&+4\lambda^2\left\|\log\pi_w^*(a|s) - \log\pi_\lambda^*(a|s)\right\|_2^2.
\end{aligned}$$

Proof Specifically, due to the strongly convexity of square function, we have

$$\begin{aligned}\ell(\widehat{V}_w^*,\widehat{\pi}_w^*)-\ell(V_\lambda^*,\pi_\lambda^*) &= 2\mathbb{E}\left[\bar{\Delta}_{V_\lambda^*,\pi_\lambda^*}(s,a)\left(\bar{\Delta}_{\widehat{V}_w^*,\widehat{\pi}_w^*}(s,a)-\bar{\Delta}_{V_\lambda^*,\pi_\lambda^*}(s,a)\right)\right]\\ &+ \mathbb{E}_{\mu\pi_b}\left[\left(\bar{\Delta}_{\widehat{V}_w^*,\widehat{\pi}_w^*}(s,a)-\bar{\Delta}_{V_\lambda^*,\pi_\lambda^*}(s,a)\right)^2\right]\\ &\geqslant \int\left(\bar{\Delta}_{\widehat{V}_w^*,\widehat{\pi}_w^*}(s,a)-\bar{\Delta}_{V_\lambda^*,\pi_\lambda^*}(s,a)\right)^2\mu(s)\pi_b(a|s)dsda\\ &:= \left\|\bar{\Delta}_{\widehat{V}_w^*,\widehat{\pi}_w^*}(s,a)-\bar{\Delta}_{V_\lambda^*,\pi_\lambda^*}(s,a)\right\|_2^2,\end{aligned}$$

where $\Delta(s,a,s')=R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)-V(s)$ and the second inequality is because the optimality of V_λ^* and π_λ^*. Therefore, we have

$$\begin{aligned}&\sqrt{\ell(\widehat{V}_w^*,\widehat{\pi}_w^*)-\ell(V_\lambda^*,\pi_\lambda^*)}\geqslant\left\|\bar{\Delta}_{\widehat{V}_w^*,\widehat{\pi}_w^*}(s,a)-\bar{\Delta}_{V_\lambda^*,\pi_\lambda^*}(s,a)\right\|_2\\ \geqslant&\left|\left\|\gamma\mathbb{E}_{s'|s,a}\left[\widehat{V}_w^*(s')-V_\lambda^*(s')\right]-\left(\widehat{V}_w^*(s)-V_\lambda^*(s)\right)\right\|_2-\lambda\left\|\log\widehat{\pi}_w^*(a|s)-\log\pi_\lambda^*(a|s)\right\|_2\right|\\ =&\left|\left\|\widehat{V}_w^*-V_\lambda^*\right\|_{\mu\pi_b}-\lambda\left\|\log\widehat{\pi}_w^*(a|s)-\log\pi_\lambda^*(a|s)\right\|_2\right|\end{aligned}$$

which implies

$$\begin{aligned}\left\|\widehat{V}_w^*-V_\lambda^*\right\|_{\mu\pi_b}^2 &\leqslant 2\left(\ell(\widehat{V}_w^*,\widehat{\pi}_w^*)-\ell(V_\lambda^*,\pi_\lambda^*)\right)+2\lambda^2\left\|\log\widehat{\pi}_w^*(a|s)-\log\pi_\lambda^*(a|s)\right\|_2^2\\ &\leqslant 2\left(\ell(\widehat{V}_w^*,\widehat{\pi}_w^*)-\ell(V_\lambda^*,\pi_\lambda^*)\right)+4\lambda^2\left\|\log\widehat{\pi}_w^*(a|s)-\log\pi_w^*(a|s)\right\|_2^2\\ &\quad+4\lambda^2\left\|\log\pi_w^*(a|s)-\log\pi_\lambda^*(a|s)\right\|_2^2.\end{aligned}$$

■

In regular MDP with Assumption 10, with appropriate C, this constraint does not introduce any loss. We denote the family of value functions and policies by parametrization as $\mathcal{V}_w$, $\mathcal{P}_w$, respectively. Then, for V and $\log\pi$ uniformly bounded by $C_\infty=\max\left\{\frac{C_R}{1-\gamma},C_\pi\right\}$ and the square loss is uniformly K-Lipschitz continuous, by Proposition 23, we have

Corollary 58 *$\ell(V,\pi)-\ell_w(V,\pi)\leqslant(K+C_\infty)\epsilon_{app}^\nu$ where $\epsilon_{app}=\sup_{\nu\in\mathcal{C}}\inf_{h\in\mathcal{H}}\|\nu-h\|_\infty$ with $\mathcal{C}$ denoting the Lipschitz continuous function space and $\mathcal{H}$ denoting the hypothesis space.*

Proof Denote the $\phi(V,\pi,\nu) := \mathbb{E}_{s,a,s'}\left[\nu(s,a)\left(R(s,a)+\gamma V(s')-\lambda\log\pi(a|s)-V(s)\right)\right] - \frac{1}{2}\mathbb{E}_{s,a,s'}\left[\nu^2(s,a)\right]$, we have $\phi(V,\pi,\nu)$ is $(K+C_\infty)$-Lipschitz continuous w.r.t. $\|\cdot\|_\infty$. Denote $\nu^*_{V,\pi} = \text{argmax}_\nu\,\phi(V,\pi,\nu)$, $\nu^{\mathcal{H}}_{V,\pi} = \text{argmax}_{\nu\in\mathcal{H}}\,\phi(V,\pi,\nu)$, and $\hat{\nu}_{V,\pi} = \min_{\nu\in\mathcal{H}}\left\|\nu-\nu^*_{V,\pi}\right\|_\infty$

$$\begin{aligned}\ell(V,\pi)-\ell_w(V,\pi) &= \phi(V,\pi,\nu^*_{V,\pi})-\phi(V,\pi,\nu^{\mathcal{H}}_{V,\pi})\\ &\leqslant \phi(V,\pi,\nu^*_{V,\pi})-\phi(V,\pi,\hat{\nu}_{V,\pi}) \leqslant (K+C_\infty)\epsilon^\nu_{\text{app}}.\end{aligned}$$

■

For the third term in Lemma 57, we have

$$\begin{aligned}\lambda\left\|\log\pi^*_w(a|s)-\log\pi^*_\lambda(a|s)\right\|_2^2 &\leqslant \ell\left(V,\pi^*_w\right)-\ell\left(V,\pi^*_\lambda\right) \qquad \text{(D.7)}\\ &= \ell_w\left(V,\pi^*_w\right)-\ell_w\left(V,\pi^*_\lambda\right)+\left(\ell\left(V,\pi^*_w\right)-\ell_w\left(V,\pi^*_w\right)\right)\\ &\quad -\left(\ell\left(V,\pi^*_\lambda\right)-\ell_w\left(V,\pi^*_\lambda\right)\right)\\ &\leqslant C_\nu\inf_{\pi_w}\left\|\lambda\log\pi_w-\lambda\log\pi^*_\lambda\right\|_\infty+2(K+C_\infty)\epsilon^\nu_{\text{app}}\\ &\leqslant C_\nu\epsilon^\pi_{\text{app}}(\lambda)+2(K+C_\infty)\epsilon^\nu_{\text{app}}\end{aligned}$$

where $C_\nu = \max_{\nu\in\mathcal{H}_w}\|\nu\|_2$. The first inequality comes from the strongly convexity of $\ell(V,\pi)$ w.r.t. $\lambda\log\pi$, the second inequality comes from Section 5 in [25] and Corollary 58 with $\epsilon^\pi_{\text{app}}(\lambda) := \sup_{\pi\in\mathcal{P}_\lambda}\inf_{\pi_w\in\mathcal{P}_w}\left\|\lambda\log\pi_w-\lambda\log\pi\right\|_\infty$ with

$$\mathcal{P}_\lambda := \left\{\pi\in\mathcal{P}, \pi(a|s) = \exp\left(\frac{Q(s,a)-\mathcal{L}(Q)}{\lambda}\right), \|Q\|_2 \leqslant C_V\right\}.$$

Based on the derivation of $\mathcal{P}_\lambda$, with continuous $\mathcal{A}$, it can be seen that as $\lambda\to 0$,

$$\mathcal{P}_0 = \left\{\pi\in\mathcal{P}, \pi(a|s) = \delta_{a_{\max}(s)}(a)\right\},$$

which results $\epsilon^\pi_{\text{app}}(\lambda)\to\infty$, and as λ increasing as finite, the policy becomes smoother, resulting smaller approximate error in general. With discrete $\mathcal{A}$, although the $\epsilon^\pi_{\text{app}}(0)$ is bounded, the approximate error still decreases as λ increases. The similar correspondence

also applies to $\epsilon^V_{\text{app}}(\lambda)$. The concrete correspondence between λ and $\epsilon_{\text{app}}(\lambda)$ depends on the specific form of the function approximators, which is an open problem and out of the scope of this dissertation.

For the second term in 57,

$$\lambda \left\|\log \widehat{\pi}^*_w(a|s) - \log \pi^*_w(a|s)\right\|_2 \leqslant \lambda \left\|\log \widehat{\pi}^*_w(a|s)\right\|_2 + \lambda \left\|\log \pi^*_w(a|s)\right\|_2 \leqslant 2\lambda C_\pi. \tag{D.8}$$

For the first term, we have

$$\begin{aligned}
& \ell(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell(V^*_\lambda, \pi^*_\lambda) \qquad (D.9)\\
= \;& \ell(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell_w(\widehat{V}^*_w, \widehat{\pi}^*_w) + \ell_w(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell_w(V^*_\lambda, \pi^*_\lambda) + \ell_w(V^*_\lambda, \pi^*_\lambda) - \ell(V^*_\lambda, \pi^*_\lambda)\\
\leqslant \;& 2(K + C_\infty)\epsilon^\nu_{\text{app}} + \ell_w(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell_w(V^*_\lambda, \pi^*_\lambda)\\
= \;& 2(K + C_\infty)\epsilon^\nu_{\text{app}} + \ell_w(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell_w(V^*_w, \pi^*_w) + \ell_w(V^*_w, \pi^*_w) - \ell_w(V^*_\lambda, \pi^*_\lambda)\\
\leqslant \;& 2(K + C_\infty)\epsilon^\nu_{\text{app}} + C_\nu \left((1+\gamma)\epsilon^V_{\text{app}}(\lambda) + \epsilon^\pi_{\text{app}}(\lambda)\right) + \ell_w(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell_w(V^*_w, \pi^*_w).
\end{aligned}$$

The last inequality is because

$$\begin{aligned}
\ell_w(V^*_w, \pi^*_w) - \ell_w(V^*_\lambda, \pi^*_\lambda) &= \inf_{V_w, \pi_w} \ell_w(V_w, \pi_w) - \ell_w(V^*_\lambda, \pi^*_\lambda)\\
&\leqslant C_\nu \inf_{V_w, \pi_w} \left((1+\gamma)\left\|V_w - V^*_\lambda\right\|_\infty + \lambda \left\|\log \pi_w - \log \pi^*_\lambda\right\|_\infty\right)\\
&\leqslant C_\nu \left((1+\gamma)\epsilon^V_{\text{app}}(\lambda) + \epsilon^\pi_{\text{app}}(\lambda)\right),
\end{aligned}$$

where the second inequality comes from Section 5 in [25].

Combine (D.7), (D.8) and (D.9) into Lemma 57 and Lemma 56 together with (D.6), we achieve

Lemma 59 (Error decomposition)

$$\begin{aligned}
\left\|\widehat{V}^*_w - V^*\right\|^2_{\mu\pi_b} \leqslant\; & \underbrace{2\left(4(K + C_\infty)\epsilon^\nu_{app} + C_\nu(1+\gamma)\epsilon^V_{app}(\lambda) + 3C_\nu\epsilon^\pi_{app}(\lambda)\right)}_{\textit{approximation error due to parametrization}}\\
& + \underbrace{16\lambda^2 C^2_\pi + \left(2\gamma^2 + 2\right)\left(\frac{\gamma\lambda}{1-\gamma}\max_{\pi\in\mathcal{P}} H(\pi)\right)^2}_{\textit{bias due to smoothing}} + \underbrace{2\left(\ell_w(\widehat{V}^*_w, \widehat{\pi}^*_w) - \ell_w(V^*_w, \pi^*_w)\right)}_{\textit{statistical error}}.
\end{aligned}$$

We can see that the bound includes the errors from three aspects: **i)**, the approximation error induced by parametrization of V, π, and ν; **ii)**, the bias induced by smoothing technique; **iii)**, the statistical error. As we can see from Lemma 59, λ plays an important role in balance the approximation error and smoothing bias.

D.4.2 Statistical Error

In this section, we analyze the generalization error. For simplicity, we denote the T finite-sample approximation of

$$\begin{aligned} L(V,\pi,\nu) &= \mathbb{E}[\phi_{V,\pi,\nu}(s,a,R,s')] \\ &:= \mathbb{E}\left[2\nu(s,a)\left(R(s,a)+\gamma V(s')-V(s)-\lambda\log\pi(a|s)\right)-\nu^2(s,a)\right], \end{aligned}$$

as

$$\begin{aligned} \widehat{L}_T(V,\pi,\nu) &= \frac{1}{T}\sum_{i=1}^{T}\phi_{V,\pi,\nu}(s,a,R,s') \\ &:= \frac{1}{T}\sum_{i=1}^{T}\left(2\nu(s_i,a_i)\left(R(s_i,a_i)+\gamma V(s_i')-V(s_i)-\lambda\log\pi(a_i|s_i)\right)-\nu^2(s_i,a_i)\right), \end{aligned}$$

where the samples $\{(s_i,a_i,s_i',R_i)\}_{i=0}^{T}$ are sampled *i.i.d.* or from β-mixing stochastic process.

By definition, we have,

$$\begin{aligned} & \ell_w(\widehat{V}_w^*,\widehat{\pi}_w^*)-\ell_w(V_w^*,\pi_w^*) \\ = & \max_{\nu\in\mathcal{H}_w} L_w\left(\widehat{V}_w^*,\widehat{\pi}_w^*,\nu\right)-\max_{\nu\in\mathcal{H}_w} L_w\left(V_w^*,\hat{\pi}_w^*,\nu\right) \\ = & L_w\left(\widehat{V}_w^*,\widehat{\pi}_w^*,\nu_w\right)-L_w\left(V_w^*,\hat{\pi}_w^*,\nu_w\right)+\underbrace{L_w\left(V_w^*,\hat{\pi}_w^*,\nu_w\right)-\max_{\nu\in\mathcal{H}_w} L_w\left(V_w^*,\hat{\pi}_w^*,\nu\right)}_{\leqslant 0} \\ \leqslant & L_w\left(\widehat{V}_w^*,\widehat{\pi}_w^*,\nu_w\right)-L_w\left(V_w^*,\hat{\pi}_w^*,\nu_w\right) \\ \leqslant & 2\sup_{V,\pi,\nu\in\mathcal{F}_w\times\mathcal{P}_w\times\mathcal{H}_w}\left|\widehat{L}_T\left(V,\pi,\nu\right)-L_w\left(V,\pi,\nu\right)\right| \end{aligned}$$

where $\nu_w=\max_{\nu\in\mathcal{H}_w} L_w\left(\widehat{V}_w^*,\widehat{\pi}_w^*,\nu\right)$.

The latter can be bounded by covering number or Rademacher complexity on hypothesis

space $\mathcal{F}_w \times \mathcal{P}_w \times \mathcal{H}_w$ with rate $\mathcal{O}\left(\sqrt{\frac{\log T}{T}}\right)$ with high probability if the samples are *i.i.d.* or from β-mixing stochastic processes [169].

We will use a generalized version of Pollard's tail inequality to β-mixing sequences, *i.e.*,

Lemma 60 *[Lemma 5, [169]] Suppose that $z_1, \ldots, Z_N \in \mathcal{Z}$ is a stationary β-mixing process with mixing coefficient $\{\beta_m\}$ and that $\mathcal{G}$ is a permissible class of $\mathcal{Z} \to [-C, C]$ functions, then,*

$$\mathbb{P}\left(\sup_{g\in\mathcal{G}}\left|\frac{1}{N}\sum_{i=1}^{N} g(Z_i) - \mathbb{E}\left[g(Z_1)\right]\right| > \epsilon\right) \leqslant 16\mathbb{E}\left[\mathcal{N}_1\left(\frac{\epsilon}{8}, \mathcal{G}, (Z_i'; i \in H)\right)\right]\exp\left(\frac{-m_N\epsilon^2}{128C^2}\right) + 2m_N\beta_{k_N+1},$$

where the "ghost" samples $Z_i' \in \mathcal{Z}$ and $H = \cup_{j=1}^{m_N} H_i$ which are defined as the blocks in the sampling path.

The covering number is highly related to pseudo-dimension, *i.e.*,

Lemma 61 *[Corollary 3, [191]] For any set $\mathcal{X}$, any points $x^{1:N} \in \mathcal{X}^N$, any class $\mathcal{F}$ of functions on $\mathcal{X}$ taking values in $[0, C]$ with pseudo-dimension $D_{\mathcal{F}} < \infty$, and any $\epsilon > 0$,*

$$\mathcal{N}\left(\epsilon, \mathcal{F}, x^{1:N}\right) \leqslant e\left(D_{\mathcal{F}} + 1\right)\left(\frac{2eC}{\epsilon}\right)^{D_{\mathcal{F}}}$$

Once we have the covering number of $\Phi(V, \pi, \nu)$, plug it into lemma 60, we will achieve the statistical error,

Theorem 29 (Stochastic error) *Under Assumption 11, with at least probability $1 - \delta$,*

$$\ell_w(\widehat{V}_w^*, \widehat{\pi}_w^*) - \ell_w(V_w^*, \pi_w^*) \leqslant 2\sqrt{\frac{M\left(\max\left(M/b, 1\right)\right)^{1/\kappa}}{C_2 T}},$$

where $M = \frac{D}{2}\log t + \log\left(e/\delta\right) + \log^+\left(\max\left(C_1 C_2^{D/2}, \bar{\beta}\right)\right)$.

Proof We use lemma 60 with $\mathcal{Z} = \mathcal{S} \times \mathcal{A} \times \mathbb{R} \times \mathcal{S}$ and $\mathcal{G} = \phi_{\mathcal{F}_w \times \mathcal{P}_w \times \mathcal{H}_w}$. For $\forall \Phi(V, \pi, \nu) \in \mathcal{G}$,

it is bounded by $C = \frac{2}{1-\gamma} C_R + \lambda C_\pi$. Thus,

$$\mathbb{P}\left(\sup_{V,\pi,\nu \in \mathcal{F}_w \times \mathcal{P}_w \times \mathcal{H}_w} \left| \frac{1}{T} \sum_{i=1}^{T} \phi_{V,\pi,\nu}\left((s,a,s',R)_i\right) - \mathbb{E}\left[\phi_{V,\pi,\nu}\right] \right| \geqslant \epsilon/2 \right) \tag{D.10}$$

$$\leqslant 16 \mathbb{E}\left[\mathcal{N}\left(\frac{\epsilon}{16}, \mathcal{G}, (Z_i'; i \in H) \right) \right] \exp\left(-\frac{m_t}{2} \left(\frac{\epsilon^2}{16C} \right)^2 \right) + 2 m_T \beta_{k_T}. \tag{D.11}$$

With some calculation, the distance in $\mathcal{G}$ can be bounded,

$$\begin{aligned} & \frac{1}{T} \sum_{i \in H} \left| \phi_{V_1,\pi_1,\nu_1}(Z_i') - \phi_{V_2,\pi_2,\nu_2}(Z_i') \right| \\ \leqslant \quad & \frac{4C}{T} \sum_{i \in H} \left| \nu_1(s_i,a_i) - \nu_2(s_i,a_i) \right| + \frac{2(1+\gamma)C}{T} \sum_{i \in H} \left| V_1(s_i) - V_2(s_i) \right| \\ & + \frac{2\lambda C}{T} \sum_{i \in H} \left| \log \pi_1(a_i|s_i) - \log \pi_2(a_i|s_i) \right|, \end{aligned}$$

which leads to

$$\mathcal{N}\left(12C\epsilon', \mathcal{G}, (Z_i'; i \in H)\right) \leqslant \mathcal{N}(\epsilon', \mathcal{F}_w, (Z_i'; i \in H)) \mathcal{N}(\epsilon', \mathcal{P}_w, (Z_i'; i \in H)) \mathcal{N}(\epsilon', \mathcal{H}_w, (Z_i'; i \in H))$$

with $\lambda \in (0, 2]$. To bound these factors, we apply lemma 61. We denote the psuedo-dimension of $\mathcal{F}_w$, $\mathcal{P}_w$, and $\mathcal{H}_w$ as D_V, D_π, and D_ν, respectively. Thus,

$$\mathcal{N}\left(12C\epsilon', \mathcal{G}, (Z_i'; i \in H)\right) \leqslant e^3 \left(D_V + 1\right)\left(D_\pi + 1\right)\left(D_\nu + 1\right) \left(\frac{4eC}{\epsilon'} \right)^{D_V + D_\pi + D_\nu},$$

which implies

$$\mathcal{N}\left(\frac{\epsilon}{16}, \mathcal{G}, (Z_i'; i \in H) \right) \leqslant e^3 \left(D_V + 1\right)\left(D_\pi + 1\right)\left(D_\nu + 1\right) \left(\frac{768eC^2}{\epsilon'} \right)^{D_V + D_\pi + D_\nu} = C_1 \left(\frac{1}{\epsilon} \right)^D,$$

where $C_1 = e^3 \left(D_V + 1\right)\left(D_\pi + 1\right)\left(D_\nu + 1\right)\left(768eC^2\right)^D$ and $D = D_V + D_\pi + D_\nu$, *i.e.*, the "effective" psuedo-dimension.

Plug this into Eq. (D.10), we obtain

$$\begin{aligned}
&\mathbb{P}\left(\sup_{V,\pi,\nu\in\mathcal{F}_w\times\mathcal{P}_w\times\mathcal{H}_w}\left|\frac{1}{T}\sum_{i=1}^{T}\phi_{V,\pi,\nu}\left((s,a,s',R)_i\right)-\mathbb{E}\left[\phi_{V,\pi,\nu}\right]\right|\geqslant\epsilon/2\right)\\
\leqslant\;& C_1\left(\frac{1}{\epsilon}\right)^{D}\exp\left(-4C_2m_t\epsilon^2\right)+2m_T\beta_{k_T},
\end{aligned}$$

with $C_2=\frac{1}{2}\left(\frac{1}{8C}\right)^2$. If $D\geqslant 2$, and $C_1,C_2,\bar{\beta},b,\kappa>0$, for $\delta\in(0,1]$, by setting $k_t=\left\lceil\left(C_2T\epsilon^2/b\right)^{\frac{1}{\kappa+1}}\right\rceil$ and $m_T=\frac{T}{2k_T}$, by lemma 14 in [169], we have

$$C_1\left(\frac{1}{\epsilon}\right)^{D}\exp\left(-4C_2m_T\epsilon^2\right)+2m_T\beta_{k_T}<\delta,$$

with $\epsilon=\sqrt{\frac{M(\max(M/b,1))^{1/\kappa}}{C_2t}}$ where $M=\frac{D}{2}\log T+\log\left(e/\delta\right)+\log^+2\left(\max\left(C_1C_2^{D/2},\bar{\beta}\right)\right)$. ■

With the statistical error bound provided in Theorem 29 for solving the derived min-max problem with arbitrary learnable nonlinear approximators using off-policy samples, we can achieve the analysis of the total error, *i.e.*,

Theorem 30 *Let $\hat{V}_w^T$ be a candidate solution output from the proposed algorithm based on off-policy samples, with at least probability $1-\delta$, we have*

$$\begin{aligned}
\left\|\widehat{V}_w^N-V^*\right\|_{\mu\pi_b}^2\;\leqslant\;&\underbrace{2\left(6(K+C_\infty)\epsilon_{app}^{\nu}+C_\nu(1+\gamma)\epsilon_{app}^{V}(\lambda)+3C_\nu\epsilon_{app}^{\pi}(\lambda)\right)}_{\textit{approximation error due to parametrization}}\\
&+\underbrace{16\lambda^2C_\pi^2+\left(2\gamma^2+2\right)\left(\frac{\gamma\lambda}{1-\gamma}\max_{\pi\in\mathcal{P}}H(\pi)\right)^2}_{\textit{bias due to smoothing}}\\
&+\underbrace{4\sqrt{\frac{M\left(\max\left(M/b,1\right)\right)^{1/\kappa}}{C_2T}}}_{\textit{statistical error}}+\underbrace{\left\|\widehat{V}_w^N-\widehat{V}_w^*\right\|_{\mu\pi_b}^2}_{\textit{optimization error}}.
\end{aligned}$$

where M is defined as above.

This theorem can be proved by combining Theorem 29 into Lemma 59.

D.4.3 Convergence Analysis

As we discussed in Section 6.5.1, the SBEED algorithm converges to a stationary point if we can achieve the optimal solution to the dual functions. However, in general, such conditions restrict the parametrization of the dual functions. In this section, we first provide the proof for Theorem 28. Then, we provide a variant of the SBEED in Algorithm 12, which still achieve the asymptotic convergence with arbitrary function approximation for the dual function, including neural networks with smooth activation functions.

Theorem 28[Convergence, [189]] *Consider the case when Euclidean distance is used in the algorithm. Assume that the parametrized objective $\widehat{\ell}_T(V_w, \pi_w)$ is K-Lipschitz and variance of its stochastic gradient is bounded by σ^2. Let the algorithm run for N iterations with stepsize $\zeta_k = \min\{\frac{1}{K}, \frac{D'}{\sigma\sqrt{N}}\}$ for some $D' > 0$ and output $w^1, \ldots, w^N$. Setting the candidate solution to be $(\widehat{V}_w^N, \widehat{\pi}_w^N)$ with w randomly chosen from $w^1, \ldots, w^N$ such that $P(w = w^j) = \frac{2\zeta_j - K\zeta_j^2}{\sum_{j=1}^N (2\zeta_j - K\zeta_j^2)}$, then it holds that $\mathbb{E}\left[\left\|\nabla \widehat{\ell}_T(\widehat{V}_w^N, \widehat{\pi}_w^N)\right\|^2\right] \leqslant \frac{KD^2}{N} + (D' + \frac{D}{D'})\frac{\sigma}{\sqrt{N}}$ where $D := \sqrt{2(\widehat{\ell}_T(V_w^1, \pi_w^1) - \min \widehat{\ell}_T(V_w, \pi_w))/K}$ represents the distance of the initial solution to the optimal solution.*

The Theorem 28 straightforwardly generalizes the convergence result in [189] to min-max optimization.

Proof As we discussed, given the empirical off-policy samples, the proposed algorithm can be understood as solving $\min_{V_w, \pi_w} \widehat{\ell}_T(V_w, \pi_w) := \widehat{L}_T(V_w, \pi_w; \nu_w^*)$ where ν_w^* denotes $\arg\max_{\nu_w} \widehat{L}_T(V_w, \pi_w; \nu_w)$.

Following the Theorem 2.1 in [189], as long as the gradients $\left[\nabla_{V_w} \widehat{\ell}_T(V_w, \pi_w),\ \nabla_{\pi_w} \widehat{\ell}_T(V_w, \pi_w)\right]$ are unbiased, under the provided conditions, the finite-step convergence rate can be obtained. The unbiasedness of the gradient estimator is already proved in Theorem 27.

■

Next, we will show that in the setting that off-policy samples are given, under some mild conditions on the neural networks parametrization, the Algorithm 12 will achieve a local Nash equilibrium of the empirical objective asymptotically, *i.e.*, $\left(w_V^+, w_\pi^+, w_\varsigma^+\right)$, such

Algorithm 12 A variant of SBEED learning

Initialize $w = (w_V, w_\pi, w_\varsigma)$ and π_b randomly, set ϵ.
for episode $i = 1, \ldots, T$ **do**
  for size $k = 1, \ldots, K$ **do**
    Add new transition (s, a, r, s') into $\mathcal{D}$ by executing behavior policy π_b.
  end for
  for iteration $j = 1, \ldots, N$ **do**
    Sample mini-batch $\{s, a, s'\}^m \sim \mathcal{D}$.
    Compute the stochastic gradient w.r.t. w_ς as $G_\varsigma = -\frac{1}{m}\sum_{\{s,a,s'\}\sim\mathcal{D}} \left(\delta(s, a, s') - \varsigma(s, a)\right) \nabla_{w_\varsigma} \varsigma(s, a)$
    Compute the stochastic gradients w.r.t. w_V and w_π as (27) with w_ς^t, denoted as G_V and G_π, respectively.
    Decay the stepsize ξ_j and ζ_j.
    Update the parameters of primal function by solving the prox-mappings, *i.e.*,

$$\begin{aligned} \text{update } \varsigma: &\quad w_\varsigma^j = P_{w_\varsigma^{j-1}}(-\xi_j G_\varsigma) \\ \text{update } V: &\quad w_V^j = P_{w_V^{j-1}}(\zeta_j G_V) \\ \text{update } \pi: &\quad w_\pi^j = P_{w_\pi^{j-1}}(\zeta_j G_\pi) \end{aligned}$$

  end for
  Update behavior policy $\pi_b = \pi^N$.
end for

that

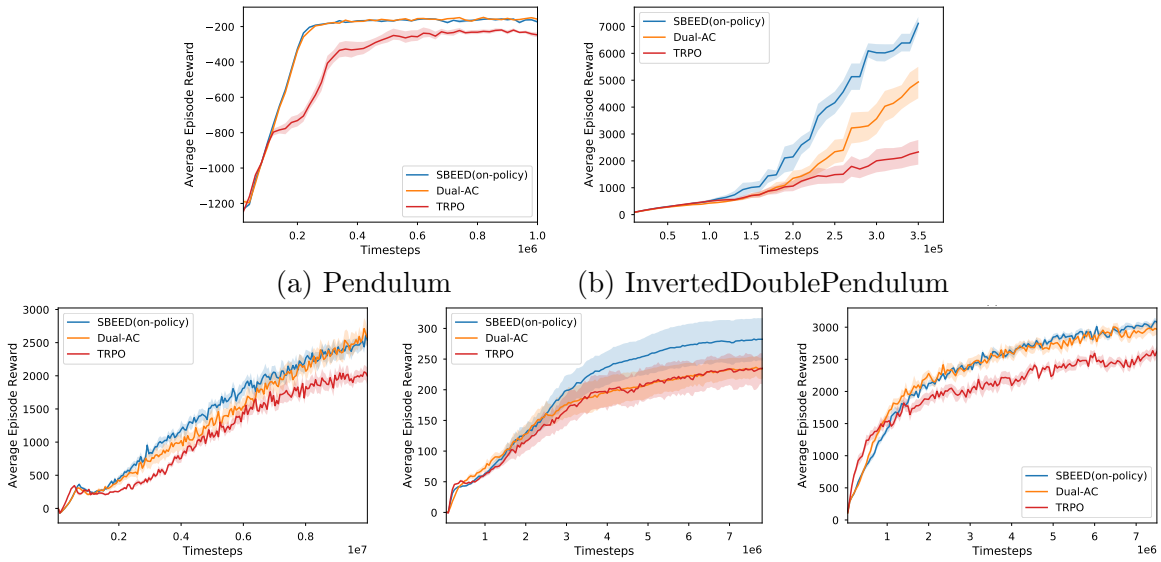
$$\nabla_{w_V, w_\pi} \widehat{L}_\eta \left(w_V^+, w_\pi^+, w_\varsigma^+\right) = 0, \quad \nabla_{w_\varsigma} \widehat{L}_\eta \left(w_V^+, w_\pi^+, w_\varsigma^+\right) = 0.$$

In fact, by applying different decay rate of the stepsizes appropriately for the primal and dual variables in the two time scales updates, the asymptotic convergence of the Algorithm 12 to local Nash equilibrium can be easily obtained by applying the Theorem 1 in [225], which is original provided by [226]. We omit the proof which is not the major contribution of this thesis. Please refer to [225, 226] for further details.

D.5 Additional Experiments

D.5.1 On-policy Comparison in Continuous Control Tasks

We compared the SBEED to TRPO and Dual-AC in on-policy setting. We followed the same experimental set up as it is in off-policy setting. We ran the algorithm with 5 random seeds and reported the average rewards with 50% confidence intervals. The empirical comparison results are illustrated in Figure D.1. We can see that in all these tasks, the proposed



(a) Pendulum (b) InvertedDoublePendulum

(c) HalfCheetah (d) Swimmer (e) Hopper

Figure D.1: The results of SBEED against TRPO and Dual-AC in the on-policy setting. Each plot shows average reward during training across 5 random runs, with 50% confidence interval. The x-axis is the number of training iterations. SBEED achieves better or comparable performance than TRPO and Dual-AC on all tasks.

SBEED achieves significantly better performance than the other algorithms. This can be thought as another ablation study that we switch off the "off-policy" in our algorithm. The empirical results demonstrate that the proposed algorithm is more flexible to way of the data sampled.

We set the step size to be 0.01 and the batch size to be 52 trajectories in each iteration in all algorithms in the on-policy setting. For TRPO, the CG damping parameter is set to be 10^{-4}.

REFERENCES

[1] A. Zellner, "Optimal Information Processing and Bayes's Theorem," *The American Statistician*, vol. 42, no. 4, Nov. 1988.

[2] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[3] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., Cambridge, MA: MIT Press, 2002.

[4] B. Póczos, A. Singh, A. Rinaldo, and L. A. Wasserman, "Distribution-free distribution regression.," in *AISTATS*, 2013, pp. 507–515.

[5] Z. Szabó, B. Sriperumbudur, B. Póczos, and A. Gretton, "Learning theory for distribution regression," *Journal of Machine Learning Research*, vol. 17, no. 152, pp. 1–40, 2016.

[6] Y. LeCun, *MNIST handwritten digit database*, 1998.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] J. C. Platt, "Sequential minimal optimization: a fast algorithm for training support vector machines," Microsoft Research, Tech. Rep. MSR-TR-98-14, 1998.

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.

[11] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M. Balcan, and L. Song, "Scalable kernel methods via doubly stochastic gradients," in *Neural Information Processing Systems*, 2014.

[12] B. Dai, N. He, H. Dai, and L. Song, "Provable bayesian inference via particle mirror descent," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 985–994.

[13] B. Dai, N. He, Y. Pan, B. Boots, and L. Song, "Learning from conditional distributions via dual embeddings," *CoRR*, vol. abs/1607.04579, 2016.

[14] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song, "Sbeed: convergent reinforcement learning with nonlinear function approximation," *CoRR*, vol. abs/1712.10285, 2017. arXiv:1712.10285.

[15] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.

[16] B. Schölkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[17] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[18] A. Gretton, O. Bousquet, A. J. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms," in *Proceedings of the International Conference on Algorithmic Learning Theory*, S. Jain, H. U. Simon, and E. Tomita, Eds., Springer-Verlag, 2005, pp. 63–77.

[19] H. König, *Eigenvalue Distribution of Compact Operators*. Basel: Birkhäuser, 1986.

[20] A. Devinatz, "Integral representation of pd functions," *Trans. AMS*, vol. 74, no. 1, pp. 56–77, 1953.

[21] M. Hein and O. Bousquet, "Kernels, associated structures, and generalizations," Max Planck Institute for Biological Cybernetics, Tech. Rep. 127, 2004.

[22] B. Simon, *Trace ideals and their applications*, 120. American Mathematical Soc., 2010.

[23] F. R. Bach, "On the equivalence between quadrature rules and random features," *CoRR*, vol. abs/1502.06800, 2015.

[24] F. Cucker and S. Smale, "Best choices for regularization parameters in learning theory: on the bias–variance problem," *Foundations of Computational Mathematics*, vol. 2, no. 4, pp. 413–428, 2002.

[25] F. R. Bach, "Breaking the curse of dimensionality with convex neural networks," *CoRR*, vol. abs/1412.8690, 2014.

[26] P. M. Williams, "Bayesian conditionalisation and the principle of minimum information," *British Journal for the Philosophy of Science*, vol. 31, no. 2, pp. 131–144, 1980.

[27] Y. Mroueh, S. Voinea, and T. A. Poggio, "Learning with group invariant features: a kernel perspective.," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 1558–1566.

[28] L. Baird, "Residual algorithms: reinforcement learning with function approximation," in *Proc. Intl. Conf. Machine Learning*, Morgan Kaufmann, 1995, pp. 30–37.

[29] R. S. Sutton, H. R. Maei, and C. Szepesvri, "A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008, pp. 1609–1616.

[30] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[31] X. Nguyen, M. Wainwright, and M. Jordan, "Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization," in *Advances in Neural Information Processing Systems 20*, Cambridge, MA: MIT Press, 2008, pp. 1089–1096.

[32] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, "Predicting time series with support vector machines," in *Artificial Neural Networks ICANN'97*, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds., ser. Lecture Notes in Comput. Sci. Vol. 1327, Berlin: Springer-Verlag, 1997, pp. 999–1004.

[33] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds., Cambridge, MA: MIT Press, 1997, pp. 155–161.

[34] B. Schölkopf, P. Simard, A. J. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," in *Advances in Neural Information Processing Systems 10*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., Cambridge, MA: MIT Press, 1998, pp. 640–646.

[35] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for span categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048–1054, 1999.

[36] B. Schölkopf, R. C. Williamson, A. J. Smola, and J. Shawe-Taylor, "Single-class support vector machines," in *Unsupervised Learning*, J. Buhmann, W. Maass, H. Ritter, and N. Tishby, Eds., ser. Dagstuhl-Seminar-Report 235, 1999, pp. 19–20.

[37] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, pp. 389–422, 2002.

[38] A. B. Graf, A. J. Smola, and S. Borer, "Classification in a normalized feature space using support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 597–605, 2003.

[39] B. Schölkopf, A. J. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Max-Planck-Institut für biologische Kybernetik, Tech. Rep. 44, 1996.

[40] —, "Kernel principal component analysis," in *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., Cambridge, MA: MIT Press, 1999, pp. 327–352.

[41] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds., MIT Press, 1999, pp. 536–542.

[42] S. Mika, A. J. Smola, and B. Schölkopf, "An improved training algorithm for kernel Fisher discriminants," in *Artificial Intelligence and Statistics*, T. Jaakkola and T. Richardson, Eds., Also: Microsoft Research TR-2000-77, San Francisco, CA: Morgan Kaufmann Publishers, 2001, pp. 98–104.

[43] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.

[44] Y. Altun, T. Hofmann, and A. J. Smola, "Gaussian process classification for segmenting and annotating sequences," in *Proc. Intl. Conf. Machine Learning*, New York, NY: ACM Press, 2004, pp. 25–32.

[45] J. Ham, D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proceedings of the Twenty-First International Conference on Machine Learning*, New York, NY, USA: ACM Press, 2004, pp. 369–376.

[46] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. Intl. Conf. Machine Learning*, Banff, AL: ACM Press, 2004, ISBN: 1-58113-828-5.

[47] A. J. Smola, A. Gretton, L. Song, and B. Schölkopf, "A Hilbert space embedding for distributions," in *Proceedings of the International Conference on Algorithmic Learning Theory*, vol. 4754, Springer, 2007, pp. 13–31.

[48] L. Song, "Learning via Hilbert space embedding of distributions," PhD thesis, School of Information Technologies, University of Sydney, 2008.

[49] L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. J. Smola, "Hilbert space embeddings of hidden markov models," in *International Conference on Machine Learning*, 2010.

[50] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Control Systems Signals*, vol. 2, no. 4, pp. 303–314, 1989.

[51] L. Jones, "Constructive approximations for neural networks by sigmoidal functions," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1586–1589, 1990.

[52] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, Jan. 1993.

[53] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[54] C. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *Journal of Machine Learning Research*, vol. 7, pp. 2651–2667, 2006.

[55] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proceedings of the International Conference on Machine Learning*, San Francisco: Morgan Kaufmann Publishers, 2000, pp. 911–918.

[56] C. K. I. Williams and M. Seeger, "Using the Nystrom method to speed up kernel machines," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2000.

[57] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2001.

[58] P. Drineas and M. Mahoney, "On the nystr om method for approximating a gram matrix for improved kernel-based learning," *JMLR*, vol. 6, pp. 2153–2175, 2005.

[59] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *AISTATS*, 2010, pp. 113–120.

[60] A. Alaoui and M. W. Mahoney, "Fast randomized kernel ridge regression with statistical guarantees," in *Advances in Neural Information Processing Systems*, 2015, pp. 775–783.

[61] A. Rudi, R. Camoriano, and L. Rosasco, "Less is more: nyström computational regularization," in *Advances in Neural Information Processing Systems*, 2015, pp. 1657–1665.

[62] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds., Cambridge, MA: MIT Press, 2008.

[63] Q. Le, T. Sarlos, and A. J. Smola, "Fastfood — computing hilbert space expansions in loglinear time," in *International Conference on Machine Learning*, 2013.

[64] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: replacing minimization with randomization in learning," in *Neural Information Processing Systems*, 2009.

[65] D. Lopez-Paz, S. Sra, A. J. Smola, Z. Ghahramani, and B. Schlkopf, "Randomized nonlinear component analysis," 2014.

[66] T. Joachims, "Making large-scale SVM learning practical," in *Advances in kernel methods: support vector learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., MIT Press, 1999, pp. 169–184.

[67] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 567–599, 2013.

[68] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, 2004.

[69] N. Ratliff and J. Bagnell, "Kernel conjugate gradient for fast kernel machines," in *International Joint Conference on Artificial Intelligence*, vol. 20, 2007.

[70] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. on Optimization*, vol. 19, no. 4, pp. 1574–1609, Jan. 2009.

[71] P. Kar and H. Karnick, "Random feature maps for dot product kernels," in *AISTATS-12*, N. D. Lawrence and M. A. Girolami, Eds., vol. 22, 2012, pp. 583–591.

[72] N. Pham and R. Pagh, "Fast and scalable polynomial kernels via explicit feature maps," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2013, pp. 239–247.

[73] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, 2012.

[74] J. Yang, V. Sindhwani, Q. Fan, H. Avron, and M. W. Mahoney., "Random laplace feature maps for semigroup kernels on histograms," in *CVPR*, 2014.

[75] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., 2009, pp. 342–350.

[76] H. Wendland, *Scattered Data Approximation*. Cambridge, UK: Cambridge University Press, 2005.

[77] S. S. Keerthi and D. DeCoste, "A modified finite Newton method for fast solution of large scale linear SVMs," *J. Mach. Learn. Res.*, vol. 6, pp. 341–361, 2005.

[78] A. J. Smola, L. Song, and C. H. Teo, "Relative novelty detection," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 536–543.

[79] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *NIPS*, 2013, pp. 315–323.

[80] A. Agarwal, S. Kakade, N. Karampatziakis, L. Song, and G. Valiant, "Least squares revisited: scalable approaches for multi-class prediction," in *International Conference on Machine Learning (ICML)*, 2014.

[81] T. Yang, R. Jin, and S. Zhu, "On data preconditioning for regularized loss minimization," *CoRR*, 2014.

[82] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: primal estimated sub-gradient solver for SVM," in *Proc. Intl. Conf. Machine Learning*, 2007.

[83] C. D. Dang and G. Lan, "Stochastic block mirror descent methods for nonsmooth and stochastic optimization," University of Florida, Tech. Rep., 2013.

[84] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[85] A. Cotter, S. Shalev-Shwartz, and N. Srebro, "Learning optimally sparse support vector machines," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, 2013, pp. 266–274.

[86] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A. von Lilienfeld, and K.-R. Müller, "Learning invariant representations of molecules for atomization energy prediction.," in *Neural Information Processing Systems*, 2012, pp. 449–457.

[87] S. Si, C.-J. Hsieh, and I. S. Dhillon, "Memory efficient kernel approximation," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 682–713, 2017.

[88] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[89] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[90] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[91] N. Chopin, "A sequential particle filter method for static models," *Biometrika*, vol. 89, no. 3, pp. 539–551, 2002.

[92] S. Balakrishnan and D. Madigan, "A one-pass sequential monte carlo method for bayesian analysis of massive datasets," *Bayesian Analysis*, vol. 1, no. 2, pp. 345–361, Jun. 2006.

[93] M. Welling and Y.-W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *International Conference on Machine Learning (ICML)*, 2011, pp. 681–688.

[94] D. Maclaurin and R. P. Adams, *Firefly monte carlo: exact MCMC with subsets of data*, 2014.

[95] S. Ahn, A. Korattikara, and M. Welling, "Bayesian posterior sampling via stochastic gradient fisher scoring," in *International Conference on Machine Learning*, 2012.

[96] T. Chen, E. B. Fox, and C. Guestrin, "Stochastic Gradient Hamiltonian Monte Carlo," in *Proceeding of 31th International Conference on Machine Learning (ICML'14)*, 2014.

[97] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven, "Bayesian sampling using stochastic gradient thermostats," in *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., 2014, pp. 3203–3211.

[98] Y. W. Teh, A. H. Thiéry, and S. J. Vollmer, "Consistency and fluctuations for stochastic gradient Langevin dynamics," *submitted*, 2014.

[99] K. C. Z. S. J. Vollmer and Y. W. Teh, "(non-) asymptotic properties of stochastic gradient langevin dynamics," *submitted*, 2015.

[100] R. Bardenet, A. Doucet, and C. Holmes, "On markov chain monte carlo methods for tall data," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1515–1557, 2017.

[101] A. Korattikara, Y. Chen, and M. Welling, "Austerity in mcmc land: cutting the metropolis-hastings budget," in *International Conference on Machine Learning*, 2014, pp. 181–189.

[102] M. I. Jordan, Z. Gharamani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in Graphical Models*, M. I. Jordan, Ed., Kluwer Academic, 1998, pp. 105–162.

[103] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1 – 2, pp. 1–305, 2008.

[104] T. Minka, "Expectation propagation for approximative Bayesian inference," PhD thesis, MIT Media Labs, Cambridge, USA, 2001.

[105] T. Minka, "Divergence measures and message passing," Microsoft Research, Report 173, 2005.

[106] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, pp. 1303–1347, 2013.

[107] T. S. Jaakkola and M. I. Jordon, "Learning in graphical models," in, M. I. Jordan, Ed., Cambridge, MA, USA: MIT Press, 1999, ch. Improving the Mean Field Approximation via the Use of Mixture Distributions, pp. 163–173, ISBN: 0-262-60032-3.

[108] S. Gershman, M. Hoffman, and D. M. Blei, "Nonparametric variational inference," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, J. Langford and J. Pineau, Eds., New York, NY, USA: ACM, 2012, pp. 663–670.

[109] J. Zhu, N. Chen, and E. P. Xing, "Bayesian inference with posterior regularization and applications to infinite latent svms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1799–1847, Jan. 2014.

[110] J. W. Paisley, D. M. Blei, and M. I. Jordan, "Variational bayesian inference with stochastic search," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[111] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[112] B. Delyon and A. Juditsky, "On minimax wavelet estimators," *Applied and Computational Harmonic Analysis*, vol. 3, no. 3, pp. 215 –228, 1996.

[113] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *Signal Processing, IEEE Transactions on*, vol. 50, no. 3, pp. 736–746, 2002.

[114] A. Gibbs and F. Su, "On choosing and bounding probability metrics," *International Statistical Review*, vol. 70, p. 419, 2002.

[115] P. Del Moral, A. Doucet, and A. Jasra, "Sequential monte carlo samplers," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 411–436, 2006.

[116] R. M. Neal, "Defining priors for distributions using dirichlet diffusion trees," University of Toronto, Tech. Rep., 2001.

[117] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky, "Nonparametric belief propagation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.

[118] A. Ihler and D. McAllester, "Particle belief propagation," in *AISTATS*, 2009, pp. 256–263.

[119] L. Song, A. Gretton, D. Bickson, Y. Low, and C. Guestrin, "Kernel belief propagation," in *Proc. Intl. Conference on Artificial Intelligence and Statistics*, ser. JMLR workshop and conference proceedings, vol. 10, 2011.

[120] T. Lienart, Y. W. Teh, and A. Doucet, "Expectation particle belief propagation," *arXiv preprint arXiv:1506.05934*, 2015.

[121] Q. Liu and D. Wang, "Stein variational gradient descent: a general purpose bayesian inference algorithm," in *Advances in Neural Information Processing Systems*, 2016, pp. 2370–2378.

[122] F. L. Gland and N. Oudjane, "Stability and uniform approximation of nonlinear filters using the hilbert metric and application to particle filters," *The Annals of Applied Probability*, vol. 14, no. 1, pp. 144–187, 2004.

[123] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," *CoRR*, vol. abs/1309.6835, 2013.

[124] S. Patterson and Y. W. Teh, "Stochastic gradient Riemannian Langevin dynamics on the probability simplex," in *Advances in Neural Information Processing Systems*, 2013.

[125] C. Holmes and L. Held, "Bayesian auxiliary variable models for binary and multinomial regression," *Bayesian Analysis*, vol. 1, no. 1, 145-168.

[126] T Jaakkola and M. I. Jordan, "A variational approach to bayesian logistic regression models and their extensions," in *Sixth International Workshop on Artificial Intelligence and Statistics*, 1997.

[127] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[128] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[129] K. R. Canini, L. Shi, and T. L. Griff iths, "Online inference of topics with latent dirichlet allocation," in *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[130] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola, "Scalable inference in latent variable models," in *Proceedings of The 5th ACM International Conference on Web Search and Data Mining (WSDM)*, 2012.

[131] D. Mimno, M. Hoffman, and D. Blei, "Sparse stochastic inference for latent dirichlet allocation," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, J. Langford and J. Pineau, Eds., ser. ICML '12, Edinburgh, Scotland, GB: Omnipress, 2012, pp. 1599–1606, ISBN: 978-1-4503-1285-1.

[132] E. Todorov, "Linearly-solvable Markov decision problems," in *NIPS*, 2006, pp. 1369–1376.

[133] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the national academy of sciences*, vol. 106, no. 28, pp. 11 478–11 483, 2009.

[134] P. Niyogi, F. Girosi, and T. Poggio, "Incorporating prior knowledge in machine learning by creating virtual examples," *Proceedings of IEEE*, vol. 86, no. 11, pp. 2196–2209, 1998.

[135] F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio, "Unsupervised learning of invariant representations in hierarchical architectures," *arXiv preprint arXiv:1311.4158*, 2013.

[136] M. Wang, E. X. Fang, and H. Liu, "Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions," *arXiv preprint arXiv:1411.3803*, 2014.

[137] L. Song, A. Gretton, and K. Fukumizu, "Kernel embeddings of conditional distributions," *IEEE Signal Processing Magazine*, 2013.

[138] S. Grunewalder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton, and M. Pontil, "Conditional mean embeddings as regressors," in *ICML*, 2012.

[139] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009, pp. 993–1000.

[140] B. Liu, J. Liu, M. Ghavamzadeh, S. Mahadevan, and M. Petrik, "Finite-sample analysis of proximal gradient td algorithms," in *Uncertainty in Artificial Intelligence (UAI)*, AUAI Press, 2015.

[141] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of convex analysis*. Springer Science & Business Media, 2012.

[142] R. M. Rifkin and R. A. Lippert, "Value regularization and Fenchel duality," *Journal of Machine Learning Research*, vol. 8, pp. 441–479, 2007.

[143] R. DeVore, R. Howard, and C. Micchelli, "Optimal nonlinear approximation," *Manuskripta Mathematika*, 1989.

[144] B. Sriperumbudur, A. Gretton, K. Fukumizu, G. Lanckriet, and B. Schölkopf, "Injective Hilbert space embeddings of probability measures," in *Proc. Annual Conf. Computational Learning Theory*, 2008, pp. 111–122.

[145] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Springer Verlag, 1998.

[146] A. Shapiro, D. Dentcheva, et al., *Lectures on stochastic programming: modeling and theory*. SIAM, 2014, vol. 16.

[147] Z. Szabó and B. K. Sriperumbudur, "Characteristic and universal tensor product kernels," *arXiv preprint arXiv:1708.08157*, 2017.

[148] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines with selective sampling," in *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds., MIT Press, 2007, pp. 301–320.

[149] M. Hein and O. Bousquet, "Hilbertian metrics and positive definite kernels on probability measures," in *Proc. of AI & Statistics*, Z. Ghahramani and R. Cowell, Eds., vol. 10, 2005, pp. 136–143.

[150] T. Jebara, R. Kondor, and A. Howard, "Probability product kernels," *J. Mach. Learn. Res.*, vol. 5, pp. 819–844, 2004.

[151] C. Bhattacharyya, K. S. Pannagadatta, and A. J. Smola, "A second order cone programming formulation for classifying missing data," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., Cambridge, MA: MIT Press, 2005, pp. 153–160.

[152] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton, NJ: Princeton University Press, 2008.

[153] S. Grunewalder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton, "Modeling transition dynamics in MDPs with RKHS embeddings," in *ICML*, 2012.

[154] C. Dann, G. Neumann, and J. Peters, "Policy evaluation with temporal differences: a survey and comparison.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 809–883, 2014.

[155] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[156] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[157] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[158] C. J. Watkins, "Learning from delayed rewards," PhD thesis, King's College, University of Cambridge, UK, 1989.

[159] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge University Engineering Department, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[160] R. S. Sutton, "Generalization in reinforcement learning: successful examples using sparse coarse coding," in *NIPS*, 1996, pp. 1038–1044.

[161] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[162] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.

[163] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[164] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996, ISBN: 1-886529-10-8.

[165] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: safely approximating the value function," in *NIPS*, 1995, pp. 369–376.

[166] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, pp. 674–690, 1997.

[167] G. J. Gordon, "Stable function approximation in dynamic programming," in *ICML*, 1995, pp. 261–268.

[168] D. Ormoneit and Ś. Sen, "Kernel-based reinforcement learning," *Machine Learning*, vol. 49, pp. 161–178, 2002.

[169] A. Antos, C. Szepesvári, and R. Munos, "Fitted q-iteration in continuous action-space mdps," in *Advances in neural information processing systems*, 2008, pp. 9–16.

[170] J. A. Boyan, "Least-squares temporal difference learning," *Machine Learning*, vol. 49, no. 2, pp. 233–246, 2002.

[171] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *JMLR*, vol. 4, pp. 1107–1149, 2003.

[172] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, "Toward off-policy learning control with function approximation," in *ICML*, 2010, pp. 719–726.

[173] H. R. Maei, "Gradient temporal-difference learning algorithms," PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2011.

[174] S. Mahadevan, B. Liu, P. S. Thomas, W. Dabney, S. Giguere, N. Jacek, I. Gemp, and J. Liu, "Proximal reinforcement learning: a new theory of sequential decision making in primal-dual spaces," CoRR abs/1405.6757, 2014.

[175] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed, "Distributed policy evaluation under multiple behavior strategies," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1260–1274, 2015.

[176] S. S. Du, J. Chen, L. Li, L. Xiao, and D. Zhou, "Stochastic variance reduction methods for policy evaluation," in *ICML*, 2017, pp. 1049–1058.

[177] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–152, 2005.

[178] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, England: Cambridge University Press, 2004.

[179] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Robotics: Science and Systems VIII*, 2012.

[180] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *UAI*, 2016.

[181] G. Neu, A. Jonsson, and V. Gómez, "A unified view of entropy-regularized markov decision processes," *arXiv preprint arXiv:1705.07798*, 2017.

[182] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Bridging the gap between value and policy based reinforcement learning," in *NIPS*, 2017, pp. 2772–2782.

[183] K. Asadi and M. L. Littman, "An alternative softmax operator for reinforcement learning," in *ICML*, 2017, pp. 243–252.

[184] D. P. Bertsekas, *Nonlinear Programming*, 3rd. Athena Scientific, 2016, ISBN: 978-1-886529-05-2.

[185] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3–4, pp. 293–321, 1992.

[186] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization.," in *ICML*, 2015, pp. 1889–1897.

[187] S. Kakade, "A natural policy gradient," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., MIT Press, 2002, pp. 1531–1538.

[188] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade, "Towards generalization and simplicity in continuous control," *arXiv preprint arXiv:1703.02660*, 2017.

[189] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[190] M. Carrasco and X. Chen, "Mixing and moment properties of various GARCH and stochastic volatility models," *Econometric Theory*, vol. 18, no. 1, pp. 17–39, 2002.

[191] D. Haussler, "Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension," *J. Combinatorial Theory (A)*, vol. 69, no. 2, pp. 217–232, 1995, Was University of CA at Santa Cruz TR UCSC-CRL-91-41.

[192] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *ICML*, 2008, pp. 664–671.

[193] D. P. de Farias and B. Van Roy, "On the existence of fixed points for approximate value iteration and temporal-difference learning," *Journal of Optimization Theory and Applications*, vol. 105, no. 3, pp. 589–608, 2000.

[194] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," *arXiv preprint arXiv:1702.08165*, 2017.

[195] J. Rubin, O. Shamir, and N. Tishby, "Trading value and information in MDPs," *Decision Making with Imperfect Decision Makers*, pp. 57–74, 2012.

[196] Y. Liu, P. Ramachandran, Q. Liu, and J. Peng, "Stein variational policy gradient," in *UAI*, 2017.

[197] J. Schulman, P. Abbeel, and X. Chen, "Equivalence between policy gradients and soft q-learning," *arXiv preprint arXiv:1704.06440*, 2017.

[198] O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans, "Trust-PCL: an off-policy trust region method for continuous control," in *ICLR*, arXiv:1707.01891, 2018.

[199] Y. Chen and M. Wang, "Stochastic primal-dual methods and sample complexity of reinforcement learning," *arXiv preprint arXiv:1612.02516*, 2016.

[200] M. Wang, "Randomized Linear Programming Solves the Discounted Markov Decision Problem In Nearly-Linear Running Time," *ArXiv e-prints*, 2017. arXiv:1704.01869.

[201] B. Dai, A. Shaw, N. He, L. Li, and L. Song, "Boosting the actor with dual critic," *ICLR*, 2018, arXiv:1712.10282.

[202] P. J. Schweitzer and A. Seidmann, "Generalized polynomial approximations in Markovian decision processes," *Journal of Mathematical Analysis and Applications*, vol. 110, no. 2, pp. 568–582, 1985.

[203] D. P. de Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[204] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *OpenAI Gym*, arXiv:1606.01540, 2016.

[205] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: a physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 5026–5033.

[206] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv:1509.02971*, 2015.

[207] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *arXiv preprint arXiv:1709.06560*, 2017.

[208] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[209] B. Sriperumbudur, K. Fukumizu, A. Gretton, A. Hyvärinen, and R. Kumar, "Density estimation in infinite dimensional exponential families," *arXiv preprint arXiv:1312.3516*, 2013.

[210] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[211] M. A. Carreira-Perpignan and G. Hinton, "On contrastive divergence learning," in *Proc. Intl. Conference on Artificial Intelligence and Statistics*, 2005.

[212] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.

[213] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[214] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. Freitas, "Learning to learn without gradient descent by gradient descent," in *International Conference on Machine Learning*, 2017, pp. 748–756.

[215] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, 2017.

[216] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, "Generalizing hamiltonian monte carlo with neural networks," *arXiv preprint arXiv:1711.09268*, 2017.

[217] A. Rakhlin, O. Shamir, and K. Sridharan, "Making gradient descent optimal for strongly convex stochastic optimization," in *ICML*, 2012, pp. 449–456.

[218] M. P. Wand and M. C. Jones, *Kernel Smoothing*. London: Chapman and Hall, 1995.

[219] L. Devroye and L. Györfi, *Nonparametric Density Estimation: The L_1 View*. John Wiley and Sons, 1985.

[220] Q. Liu, "Stein variational gradient descent as gradient flow," in *Advances in neural information processing systems*, 2017, pp. 3118–3126.

[221] T. J. Perkins and D. Precup, "A convergent form of approximate policy iteration," in *Advances in Neural Information Processing Systems 15*, S. Thrun and K. Obermayer, Eds., Cambridge, MA: MIT Press, 2002, pp. 1595–1602.

[222] R. Munos, "Error bounds for approximate policy iteration," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 560–567.

[223] M. J. Kearns and S. P. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine Learning*, vol. 49, no. 2–3, pp. 209–232, 2002.

[224] A. L. Strehl, L. Li, and M. L. Littman, "Reinforcement learning in finite MDPs: PAC analysis," *Journal of Machine Learning Research*, vol. 10, pp. 2413–2444, 2009.

[225] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *arXiv preprint arXiv:1706.08500*, 2017.

[226] V. S. Borkar, "Stochastic approximation with two time scales," *Systems & Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.

VITA

Bo Dai is a Ph.D. candidate in the School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology. His primary research interests lie on developing effective statistical models and efficient algorithms for learning from a massive volume of complex, structured, uncertain and high-dimensional data, represented as functions, distributions, dynamics, and so on. Bo has received several recognitions for his research, including the Best Paper Award at AISTATS 2016 and the Best Paper Award at Machine Learning for Molecules and Materials workshop at NIPS 2017.